AURIX™

TC21x/TC22x/TC23x Family

32-Bit Single-Chip Microcontroller

User's Manual

V1.1 2014-12

Microcontrollers

Edition 2014-12

Published by
Infineon Technologies AG
81726 Munich, Germany

© 2015 Infineon Technologies AG
All Rights Reserved.

Legal Disclaimer

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics. With respect to any examples or hints given herein, any typical values stated herein and/or any information regarding the application of the device, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation, warranties of non-infringement of intellectual property rights of any third party.

Information

For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office (**www.infineon.com**).

Warnings

Due to technical requirements, components may contain dangerous substances. For information on the types in question, please contact the nearest Infineon Technologies Office.

Infineon Technologies components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

AURIX™

TC21x/TC22x/TC23x Family

32-Bit Single-Chip Microcontroller

User's Manual

V1.1 2014-12

Microcontrollers

TC21x/TC22x/TC23x User's Manual

Revision History: V1.1 2014-12

Significant changes since previous version: TC22x/TC23x Family User's Manual V1.0, 2014-01

Chapter	Subjects (major changes since last revision)
–	• "TC21x" added to headlines and title.
Introduction	Introduction, Table "Overview of TC22x/TC23x Family Devices": • Added LFBGA-292 package option to TC23x-ADAS. • Added TQFP-144 package option to TC21x. • Added SAL package option to all devices.
SCU/CCU	• Removed documentation of internal oscillator load caps ("adjustable Load Caps") as these are not implemented.
SCU/SCU	• Corrected DTSCON.CAL width. • Removed design related register reference from "OCDS Behavior of WDTs".
MTU	• MCONTROL.Res4 field (bits 14:12) marked as "Must be written with 0x4" in to maintain redundancy. • Sections "Reading a Single Memory Location" and "Writing a Single Memory Location" should always write MCONTROL with the MS nibble 0x4 instead of 0x0. • Design related register reference "RDR" removed.

TC21x/TC22x/TC23x User's Manual

Revision History: V1.1 2014-12

SMU	• Corrected register long name of SMU_RTC to "Recovery Timer Configuration". • DMA alarm ALM2[24] is now reserved. Alarm implemented by DMA SRI error detection alarms. • Improved description of alarm ALM3[27]. • SCU alarm ALM3[28] is now reserved. Feature not implemented. • Pre-alarms PreAlarm[129=DMA.(Safety FF Error Detection)] and PreAlarm[130=PMU.(Safety FF Error Detection)] removed because DMA and PMU do not implement Safety FF. • Added configuration hints to SMU_AG<n>CF<m> Registers: reserved alarms shall be configured as "No Action". • Enhanced description of register SMU_RMEF and SMU_RMSTS. • Enhanced description of register SMU_PCTL with advice how to operate it in the application. • Changed section "SMU Integration Guidelines". • Explained "Register Monitor" = "Safety Flip-Flop". • Improved Figure "SMU State Machine". • Improved section "FSP Fault State".

TC21x/TC22x/TC23x User's Manual

Revision History: V1.1 2014-12

PMU/Flash	• Repeated the advice to handle PVER as signal to jump to the next word-line in the receipe for the Robust EEPROM emulation. Previously this advice was only contained in the section "Handling Errors During Operation". • Showed the ECC decoder of the BootROM in the block diagram and described its SEC-DED capability. • Highlighted in the description of FCON and the wait cycle description that after System Resets and Power-On Resets the wait cycles are only sufficient for 100 MHz. • Noted in the "Application Hints" that the recommended sequence for issuing "Verify Erased Logical Sector Range" is analog to "Erase". • Changed recommendations in the "Application Hints" for ECC test patterns. Made the all-0/all-1 and address error patterns optional. Recommended to store 2 of the 4 pattern sets inverse to the other 2. • Improved description of "Load Page" to make the offsets used for 32-bit transfers more explicit. • Added to the description of "Write Page" the clear statement that each page shall be programmed only once. • Noted that "Verify Erased Logical Sector Range" is not blocked on password containing UCBs when the protection is disabled. • Removed confusing statement that unavailable DFlash ranges can be programmed to all-1. • Noted explicitly that the UCBs 8 to 11 are read-only. • Clarified that both "Write Page" and "Write Burst" can be used to program UCBs. • Removed statement about restricted FAR capabilities. • Removed note from "Erase Logical Sector Range" that warned from erasing more than 256 KByte because of increased suspend times. The suspend times are not increasing. • Changed note for "Erase Logical Sector Range" that described that erase time can be much shorter than maximum times documented in the Data Sheet. The reason was removed as there are more dependencies. • Documented "Test Pass Marker" in UCB_IFX. • Removed term "Reset Class" and replaced by named resets. • Stated explicitly that a failed HSM boot prevents device boot.
DMA	• Added address alignment rules to address registers.

TC21x/TC22x/TC23x User's Manual

Revision History: V1.1 2014-12

IR	• Added clarification: a pending Service Request is cleared in the Service Request Node with the Acknowledge from the Service Provider. The Acknowledge does not mean that the Service Request is serviced at that time (e.g. disabled or not implemented DMA channel, Trap on CPU prevents execution of Service Request). • Added description of the ECC algorithm used for the IR internal Error Detection.
OCDS	• Replaced detailed OCDS chapter with overview chapter. OCDS and Emulation functionality is described in separate documentation.
ASCLIN	• Block "Auto Baud Rate Operation" added. • Description of register field "BRD.MEASURED" improved.
QSPI	• Removed maximum baud rates from feature list as these depend on pad speeds and PCB implementation.
MultiCAN+	• Added additional text on Section Transmitter Delay Compensation to describe more details on secondary sample point implementation. • Added on footnote of Table Minimum Operating Frequencies for CAN FD "When message objects are configured as transmit or receive FIFO structures i.e MOFCRn.MMC = 0001/0010, only the base object of 1 is counted towards the minimum frequency requirement, all other slave objects on the FIFO do not count towards the minimum frequency requirement. • Under Section "Data Frames" abbreviation added for FDF, BRS, ESI bit. • Under Section "Transceiver Delay Compensation", amended from "measured Transmitter Delay" to "measured loop delay". • All reference to EDL is renamed to FDF, register bit MOFCR.EDL is renamed to MOFCR.FDF, functionality remains the same. • Changed CAN FD descriptions to customer relevant information. • Fixed pad naming for all devices. • CRC issue of CAN FD standard documented and recommended additional software CRC.
SENT	• Changed minimum tick timer to 0.2 µs. • Changed "Extended Serial Data Frame" to "Enhanced Serial Message Frame". Updated its description.

TC21x/TC22x/TC23x User's Manual

Revision History: V1.1 2014-12

GTM	• Corrected/added wiring tables for QFP-80. • Corrected description of CLK_SEL field of GTM_DTMx_CTRL registers.
CCU6	• Changed signal name "VADC_G0BFL3" to "VADCG0BFL3" for consistency with VADC chapter (no functional change).

Trademarks

TriCore® is a trademark of Infineon Technologies AG.

Copyrights

Portions Copyright © 2013 Synopsys, Inc. Used with permission. All rights reserved. Synopsys & DesignWare are registered trademarks of Synopsys, Inc.

We Listen to Your Comments

Is there any information in this document that you feel is wrong, unclear or missing? Your feedback will help us to continuously improve the quality of this document. Please send your proposal (including a reference to this document) to:

mcdocu.comments@infineon.com

Trademarks of Infineon Technologies AG

AURIX™, C166™, CanPAK™, CIPOS™, CIPURSE™, EconoPACK™, CoolMOS™, CoolSET™, CORECONTROL™, CROSSAVE™, DAVE™, DI-POL™, EasyPIM™, EconoBRIDGE™, EconoDUAL™, EconoPIM™, EconoPACK™, EiceDRIVER™, eupec™, FCOS™, HITFET™, HybridPACK™, I²RF™, ISOFACE™, IsoPACK™, MIPAQ™, ModSTACK™, my-d™, NovalithIC™, OptiMOS™, ORIGA™, POWERCODE™; PRIMARION™, PrimePACK™, PrimeSTACK™, PRO-SIL™, PROFET™, RASIC™, ReverSave™, SatRIC™, SIEGET™, SINDRION™, SIPMOS™, SmartLEWIS™, SOLID FLASH™, TEMPFET™, thinQ!™, TRENCHSTOP™, TriCore™.

Other Trademarks

Advance Design System™ (ADS) of Agilent Technologies, AMBA™, ARM™, MULTI-ICE™, KEIL™, PRIMECELL™, REALVIEW™, THUMB™, µVision™ of ARM Limited, UK. AUTOSAR™ is licensed by AUTOSAR development partnership. Bluetooth™ of Bluetooth SIG Inc. CAT-iq™ of DECT Forum. COLOSSUS™, FirstGPS™ of Trimble Navigation Ltd. EMV™ of EMVCo, LLC (Visa Holdings Inc.). EPCOS™ of Epcos AG. FLEXGO™ of Microsoft Corporation. FlexRay™ is licensed by FlexRay Consortium. HYPERTERMINAL™ of Hilgraeve Incorporated. IEC™ of Commission Electrotechnique

Internationale. IrDA™ of Infrared Data Association Corporation. ISO™ of INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. MATLAB™ of MathWorks, Inc. MAXIM™ of Maxim Integrated Products, Inc. MICROTEC™, NUCLEUS™ of Mentor Graphics Corporation. MIPI™ of MIPI Alliance, Inc. MIPS™ of MIPS Technologies, Inc., USA. muRata™ of MURATA MANUFACTURING CO., MICROWAVE OFFICE™ (MWO) of Applied Wave Research Inc., OmniVision™ of OmniVision Technologies, Inc. Openwave™ Openwave Systems Inc. RED HAT™ Red Hat, Inc. RFMD™ RF Micro Devices, Inc. SIRIUS™ of Sirius Satellite Radio Inc. SOLARIS™ of Sun Microsystems, Inc. SPANSION™ of Spansion LLC Ltd. Symbian™ of Symbian Software Limited. TAIYO YUDEN™ of Taiyo Yuden Co. TEAKLITE™ of CEVA, Inc. TEKTRONIX™ of Tektronix Inc. TOKO™ of TOKO KABUSHIKI KAISHA TA. UNIX™ of X/Open Company Limited. VERILOG™, PALLADIUM™ of Cadence Design Systems, Inc. VLYNQ™ of Texas Instruments Incorporated. VXWORKS™, WIND RIVER™ of WIND RIVER SYSTEMS, INC. ZETEX™ of Diodes Zetex Limited.

Last Trademarks Update 2011-11-11

Table of Contents

1 Introduction . . . 1-1
1.1 System Architecture of the TC21x/TC22x/TC23x . . . 1-1
1.1.1 TC21x/TC22x/TC23x Block Diagram . . . 1-2
1.1.2 Device Variants . . . 1-3

2 Package Characteristics . . . 2-1
2.1 Feature List . . . 2-1

3 On-Chip System Buses and Bus Bridges . . . 3-1
3.1 What is new . . . 3-1
3.2 SRI Crossbar (XBar_SRI) . . . 3-3
3.2.1 Introduction . . . 3-3
3.2.1.1 XBar_SRI Features . . . 3-5
3.2.2 SRI Transactions . . . 3-5
3.2.3 SRI Op-Codes . . . 3-6
3.2.4 SRI Error Conditions . . . 3-7
3.2.5 SRI Transaction ID Error Conditions . . . 3-7
3.2.6 Operational Overview . . . 3-8
3.2.6.1 Functional Blocks . . . 3-9
3.2.7 Functional Overview . . . 3-12
3.2.7.1 Arbitration Block . . . 3-12
3.2.7.2 Default Slave . . . 3-16
3.2.7.3 Register Access Protection . . . 3-18
3.2.7.4 SRI ECC Error Handling . . . 3-19
3.2.7.5 Error Tracking Capability . . . 3-21
3.2.7.6 Debug Trigger Event Generation (OCDS Level 1) . . . 3-22
3.2.7.7 Interrupt and Debug Events of the XBar_SRI Module . . . 3-24
3.2.8 Implementation of the Cross Bar (XBar_SRI) in the TC21x/TC22x/TC23x . . . 3-25
3.2.8.1 Mapping of SRI Master Modules to XBar_SRI Master Interfaces . . . 3-26
3.2.8.2 Mapping of SRI Slave modules to XBar_SRI Slave Interfaces . . . 3-27
3.2.8.3 TC21x/TC22x/TC23x SRI Master / Slave Interconnection Matrix . . . 3-28
3.2.8.4 Connection Master-Slave in XBar_SRI . . . 3-28
3.2.9 SRI Crossbar Registers . . . 3-29
3.2.9.1 TC21x/TC22x/TC23x Control Registers . . . 3-34
3.3 Shared Resource Interconnect to FPI Bus Interface (SFI Bridge) . . . 3-65
3.3.1 Functional Overview . . . 3-65
3.4 System Peripheral Bus . . . 3-66
3.4.1 Overview . . . 3-66
3.4.2 Bus Transaction Types . . . 3-68
3.4.3 Reaction of a Busy Slave . . . 3-68
3.4.4 Address Alignment Rules . . . 3-69

3.4.5 FPI Bus Basic Operations . . . 3-69
3.5 FPI Bus Control Unit (SBCU) . . . 3-71
3.5.1 FPI Bus Arbitration . . . 3-71
3.5.1.1 Arbitration on the System Peripheral Bus . . . 3-71
3.5.1.2 Default Master . . . 3-71
3.5.1.3 Arbitration Algorithms . . . 3-71
3.5.1.4 Starvation Prevention . . . 3-73
3.5.2 FPI Bus Error Handling . . . 3-73
3.5.3 System Registers . . . 3-75
3.5.3.1 Register Access Protection (ACCEN1/0) . . . 3-76
3.5.3.2 Kernel Reset Registers (KRST1/0, KRSTCLR) . . . 3-76
3.5.3.3 Clock Control Register (CLC) . . . 3-76
3.5.3.4 OCDS Control and Status Register (OCS) . . . 3-76
3.5.4 BCU Debug Support . . . 3-77
3.5.4.1 Address Triggers . . . 3-77
3.5.4.2 Signal Status Triggers . . . 3-78
3.5.4.3 Grant Triggers . . . 3-79
3.5.4.4 Combination of Triggers . . . 3-80
3.5.4.5 BCU Breakpoint Generation Examples . . . 3-80
3.5.5 System Bus Control Unit Registers . . . 3-82
3.5.5.1 SBCU System Registers . . . 3-85
3.5.5.2 SBCU Control Registers Descriptions . . . 3-88
3.5.5.3 SBCU Error Registers Descriptions . . . 3-91
3.5.5.4 SBCU OCDS Registers Descriptions . . . 3-96
3.6 On Chip Bus Master TAG Assignments . . . 3-109
3.7 On Chip Bus Access Times . . . 3-110

4 Memory Maps . . . 4-1
4.1 How to Read the Address Maps . . . 4-2
4.2 Contents of the Segments . . . 4-4
4.3 Address Map of the On Chip Bus System . . . 4-6
4.3.1 Segments 0 to 14 . . . 4-6
4.3.2 Segment 15 . . . 4-11
4.4 Memory Module Access Restrictions . . . 4-18
4.5 Side Effects from Modules to CPU0 Data Scratch Pad SRAM (DSPR0) 4-19

5 TC21x/TC22x/TC23x BootROM Content . . . 5-1
5.1 Startup Software . . . 5-1
5.1.1 Events triggering SSW execution . . . 5-1
5.1.1.1 Power-on . . . 5-1
5.1.1.2 System reset . . . 5-2
5.1.1.3 Application reset . . . 5-2
5.1.2 Clock system during start-up . . . 5-3
5.1.3 RAM overwrite during start-up . . . 5-3

5.1.4 Boot Options Summary . . . 5-3
5.1.5 Start-up mode selection . . . 5-4
5.1.5.1 Hardware configuration . . . 5-10
5.1.5.2 Configuration by Boot Mode Index (BMI) . . . 5-10
5.1.6 Startup Software Main Flow . . . 5-13
5.1.6.1 Basic Device Settings . . . 5-13
5.1.6.2 RAMs Handling . . . 5-13
5.1.6.3 Select and Prepare Startup Modes . . . 5-14
5.1.6.4 Final Chip Settings . . . 5-15
5.1.6.5 Ending the SSW and Starting the User Code . . . 5-17
5.2 Bootstrap Loaders . . . 5-18
5.2.1 ASC Bootstrap loader . . . 5-18
5.2.2 CAN Bootstrap Loader . . . 5-18
5.2.3 Summary of Bootstrap Loader Modes . . . 5-20
5.3 Shutdown request handler . . . 5-20
5.4 Preparation before to enter Stand-by mode . . . 5-20

6 CPU Subsystem . . . 6-1
6.1 AURIX Family CPU configurations . . . 6-2
6.2 Central Processing Unit Features . . . 6-4
6.3 TC1.6P Implementation Overview . . . 6-6
6.3.1 CPU Diagram . . . 6-6
6.3.2 Instruction Fetch Unit . . . 6-6
6.3.3 Execution Unit . . . 6-7
6.3.4 General Purpose Register File . . . 6-8
6.4 TC1.6E Implementation Overview . . . 6-10
6.4.1 CPU Diagram . . . 6-10
6.4.2 Instruction Fetch Unit . . . 6-11
6.4.3 Execution Unit . . . 6-12
6.4.4 General Purpose Register File . . . 6-13
6.5 Summary of functional changes from TC1.3.1 . . . 6-14
6.6 CPU Implementation-Specific Features . . . 6-15
6.6.1 Context Save Areas / Context Operations . . . 6-15
6.6.2 Program Counter (PC) Register . . . 6-15
6.6.3 Store Buffers . . . 6-16
6.6.4 Interrupt System . . . 6-17
6.6.5 Trap System . . . 6-17
6.6.6 Memory Integrity Error Handling . . . 6-19
6.6.6.1 Program Side Memories . . . 6-19
6.6.6.2 Data Side Memories . . . 6-20
6.6.6.3 Memory Initialisation . . . 6-22
6.6.7 WAIT Instruction . . . 6-22
6.6.8 Instruction Memory Range Limitations . . . 6-22

6.6.9 Atomicity of Data Accesses 6-23
6.6.10 A11 usage 6-24
6.7 Memory Addressing 6-25
6.7.1 CSFR and SFR base Locations 6-25
6.7.2 Local and Global Addressing 6-25
6.7.3 Cache Memory Access 6-26
6.8 CPU Subsystem Registers 6-27
6.8.1 CPU Core Special Function Registers (CSFR) 6-28
6.8.1.1 Registers 6-28
6.8.2 CPU General Purpose Registers 6-40
6.8.3 CPU Memory Protection Registers 6-40
6.8.4 Temporal Protection Registers 6-40
6.8.5 FPU Registers 6-41
6.8.6 Memory Integrity Registers 6-42
6.8.6.1 Register Descriptions 6-43
6.8.7 CPU Core Debug and Performance Counter Registers 6-53
6.8.7.1 Counter Source Details 6-53
6.8.8 Summary of CSFR Reset Values and Access Modes 6-56
6.8.9 Summary of SFR Reset Values and Access modes 6-67
6.9 CPU Instruction Timing 6-71
6.9.1 Integer-Pipeline Instructions 6-72
6.9.1.1 Simple Arithmetic Instruction Timings 6-72
6.9.1.2 Multiply Instruction Timings 6-76
6.9.1.3 Multiply Accumulate (MAC) Instruction Timing 6-77
6.9.1.4 Control Flow Instruction Timing TC1.6P 6-78
6.9.1.5 Control Flow Instruction Timing TC1.6E 6-79
6.9.2 Load-Store Pipeline Instructions 6-80
6.9.2.1 Address Arithmetic Timing 6-80
6.9.2.2 CSA Control Flow Instruction Timing 6-81
6.9.2.3 Load Instruction Timing 6-81
6.9.2.4 Store Instruction Timing 6-83
6.9.3 Floating Point Pipeline Timing 6-84
6.10 Program Memory Interface (PMI) 6-85
6.10.1 TC1.6P PMI Description 6-85
6.10.1.1 TC1.6P Scratchpad RAM 6-86
6.10.1.2 TC1.6P Program Cache 6-86
6.10.1.3 TC1.6P Program Line Buffer 6-88
6.10.1.4 TC1.6P CPU Slave Interface (CPS) 6-88
6.10.2 TC1.6E PMI Description 6-89
6.10.2.1 TC1.6E Program Scratchpad RAM 6-90
6.10.2.2 TC1.6E Program Cache 6-90
6.10.2.3 TC1.6E Program Line Buffer 6-92
6.10.2.4 TC1.6E CPU Slave Interface (CPS) 6-92

6.10.3 PMI Registers . . . 6-93
6.10.3.1 PMI Register Descriptions . . . 6-93
6.11 Data Memory Interface (DMI) . . . 6-98
6.11.1 TC1.6P DMI Description . . . 6-98
6.11.1.1 TC1.6P Data Scratchpad RAM (DSPR) . . . 6-99
6.11.1.2 TC1.6P Data Cache (DCACHE) . . . 6-99
6.11.2 TC1.6E DMI Description . . . 6-101
6.11.2.1 TC1.6E Data Scratchpad RAM (DSPR) . . . 6-102
6.11.2.2 TC1.6E Data Read Buffer (DRB) . . . 6-102
6.11.3 DMI Trap Generation . . . 6-103
6.11.4 DMI Registers . . . 6-105
6.11.4.1 DMI Register Descriptions . . . 6-105
6.12 Safety Features . . . 6-111
6.12.1 SRI Data Master Address Phase Error Injection . . . 6-111
6.12.2 SRI Data Master Write Phase Error Injection . . . 6-111
6.12.3 SRI Data Slave Read Phase Error Injection . . . 6-111
6.12.4 SRI Error Capture . . . 6-112
6.12.5 SRI Safe Data Master tag . . . 6-112
6.12.6 Safety Memory Protection . . . 6-112
6.12.7 Safety Register Protection . . . 6-113
6.12.8 Lock Step Implementation . . . 6-113
6.12.9 MBIST usage recommendations . . . 6-114
6.12.10 Registers Implementing Safety Features . . . 6-115

7 Lockstep Comparator Logic (LCL) . . . 7-1
7.1 Feature List . . . 7-1
7.2 Lockstep Control . . . 7-1
7.3 Lockstep Monitoring . . . 7-1
7.4 Lockstep Self Test . . . 7-2
7.5 Lockstep Failure Signalling Test . . . 7-4
7.6 Functional Redundancy . . . 7-4
7.7 Revision History . . . 7-5

8 System Control Units . . . 8-1
8.1 Clocking and Clock Control Unit (CCU) . . . 8-2
8.1.1 Clock Sources . . . 8-2
8.1.1.1 Oscillator Circuit (OSC) . . . 8-3
8.1.1.2 Back-up Clock . . . 8-10
8.1.2 Clock Speed Upscaling . . . 8-10
8.1.2.1 Phase-Locked Loop (PLL) Module . . . 8-10
8.1.2.2 ERAY Phase-Locked Loop (PLL_ERAY) Module . . . 8-24
8.1.3 Clock Distribution . . . 8-37
8.1.3.1 Clock Control Unit . . . 8-39
8.1.4 Individual Clock Generation . . . 8-60

8.1.4.1 Clock Control Register CLC 8-60
8.1.5 Clock Monitors 8-64
8.1.5.1 Clock Monitor Registers 8-66
8.1.6 Clock Emergency Behavior 8-70
8.1.7 External Clock Output 8-70
8.1.7.1 Programmable Frequency Output for EXTCLK0 8-70
8.1.7.2 Programmable Frequency Output for EXTCLK1 8-73
8.1.7.3 Clock Output Control Register 8-75
8.1.8 Clock Generation Unit 8-79
8.1.8.1 Example Sequence 8-79
8.1.9 CCU Register Address 8-81
8.1.10 CCU Kernel Registers 8-81
8.2 Reset Control Unit (RCU) 8-83
8.2.1 Reset Operation 8-84
8.2.1.1 Overview 8-84
8.2.1.2 Reset Types 8-84
8.2.1.3 Reset Sources Overview 8-85
8.2.1.4 Warm and Cold Resets 8-85
8.2.1.5 EVR Resets and PORST 8-86
8.2.1.6 Module Reset Behavior 8-86
8.2.1.7 General Reset Operation 8-87
8.2.1.8 Reset Generation 8-88
8.2.1.9 Shutdown and Reset Delay Timeout Counter (TOUTCNT) 8-88
8.2.1.10 Reset Triggers 8-89
8.2.1.11 Debug Reset Specific Behavior 8-90
8.2.1.12 Module Resets 8-90
8.2.1.13 Reset Controller Registers 8-92
8.2.2 External Reset Sources and Indications 8-102
8.2.2.1 External Service Requests ($\overline{\text{ESRx}}$) 8-102
8.2.3 Boot Software Interface 8-114
8.2.3.1 Configuration done with Start-up 8-114
8.2.3.2 Start-up Configuration Options 8-114
8.2.3.3 Status Registers 8-115
8.2.4 NMI Trap Generation 8-118
8.2.4.1 Trap Control Registers 8-119
8.2.5 RCU Register Address 8-123
8.2.6 RCU Kernel Registers 8-123
8.3 Power Supply & Power Management Controller (PMC) 8-125
8.3.1 Power Supply and Control 8-126
8.3.1.1 Introduction 8-126
8.3.1.2 Supply Mode and Topology Selection 8-126
8.3.1.3 Linear Regulator Mode 8-130
8.3.1.4 Switch Capacitor Regulator Mode 8-130

8.3.1.5 External Supply Mode 8-133
8.3.1.6 Components and Layout 8-133
8.3.1.7 Voltage Monitoring 8-134
8.3.1.8 Sequence during Power-up and Power-down 8-138
8.3.1.9 EVR Control Registers 8-142
8.3.2 Power Management 8-160
8.3.2.1 Power Management Overview 8-160
8.3.2.2 Idle Mode 8-163
8.3.2.3 Sleep Mode 8-165
8.3.2.4 Standby Mode 8-166
8.3.2.5 Wake-up Timer (WUT) 8-170
8.3.2.6 Power Management Registers 8-173
8.3.3 Power Management Register Address 8-188
8.3.4 PMC Kernel Registers 8-188
8.4 System Control Unit (SCU) 8-191
8.4.1 External Request Unit (ERU) 8-192
8.4.1.1 Introduction 8-192
8.4.1.2 ERU Input Pin Connections 8-195
8.4.1.3 External Request Selecter Unit (ERS) 8-196
8.4.1.4 Event Trigger Logic (ETL) 8-196
8.4.1.5 Connecting Matrix 8-198
8.4.1.6 Output Gating Unit (OGU) 8-200
8.4.1.7 ERU Output Pin Connections 8-204
8.4.1.8 External Request Unit Registers 8-205
8.4.2 Lockstep CPU Configuration 8-215
8.4.2.1 Logic Monitor Control Registers 8-216
8.4.3 Die Temperature Measurement 8-221
8.4.3.1 Die Temperature Sensor Register 8-222
8.4.4 Watchdog Timers 8-227
8.4.4.1 Watchdog Timers Overview 8-227
8.4.4.2 Features of the Watchdog Timers 8-229
8.4.4.3 The Endinit Functions 8-229
8.4.4.4 Timer Operation 8-236
8.4.4.5 Watchdog Timer Registers 8-240
8.4.5 Emergency Stop Output Control 8-257
8.4.5.1 Port Triggered Emergency Stop 8-257
8.4.5.2 SMU Event Triggered Emergency Stop 8-258
8.4.5.3 Emergency Stop Register 8-259
8.4.6 LBIST Support 8-261
8.4.6.1 LBIST Control Register 8-261
8.4.7 Global Overlay Controls 8-265
8.4.7.1 Global Overlay Control 8-266
8.4.8 Miscellaneous System Control 8-271

8.4.8.1 System Control Register 8-271
8.4.8.2 Identification Registers 8-273
8.4.8.3 SCU Access Restriction Registers 8-277
8.4.9 SCU Register Address 8-279
8.4.10 SCU Kernel Registers 8-279

9 Memory Test Unit (MTU) 9-1
9.1 Memory Content Initialization 9-1
9.1.1 Non-Security Applications 9-1
9.1.2 Security Applications 9-2
9.2 Safety Notifications 9-2
9.3 Memory Test Unit (MTU) Kernel Registers 9-2
9.3.1 Descriptions 9-3
9.3.2 MTU Register Overview 9-13
9.4 Memory Controllers 9-15
9.4.1 Control and Status Interfaces 9-16
9.4.1.1 Direct CPU Interface 9-16
9.4.2 Registers 9-17
9.4.2.1 MBIST/ECC Registers 9-17
9.4.3 Safety Features 9-36
9.4.4 Operation Modes 9-39
9.4.4.1 Starting a Memory Test Sequence (example) 9-39
9.4.4.2 Getting Detailed Memory Test Results 9-39
9.4.4.3 Dumping Fail Bitmap 9-39
9.4.4.4 Filling a Memory with Defined Contents 9-40
9.4.4.5 Reading a Single Memory Location 9-40
9.4.4.6 Writing to a Single Memory Location 9-41
9.4.5 Memory Controller Register Addresses 9-42
9.4.6 Memory Controller Register Overview 9-45
9.5 ECC Implementation 9-46
9.5.1 ECC 9-46
9.5.1.1 ECC Codes 9-46
9.5.2 Address Error Detection 9-61
9.6 Implementation Section 9-61
9.6.1 Memory Control Register Implementation 9-61
9.6.1.1 MEMTEST Implementation 9-62
9.6.1.2 MEMMAP Implementation 9-67
9.6.1.3 MEMSTAT Implementation 9-69
9.6.1.4 Memory Controller Instances 9-72

10 Safety Management Unit (SMU) 10-1
10.1 Introduction 10-1
10.2 Features 10-3
10.3 Functional Overview 10-5

10.4 Functional Description 10-6
10.4.1 Reset Types 10-6
10.4.2 Interfaces Overview 10-8
10.4.2.1 Interfaces to SCU 10-8
10.4.2.2 Interfaces to the Interrupt Router 10-8
10.4.2.3 Interface to the Ports (Error Pin) 10-8
10.4.2.4 Interface to the Safety Flip-Flop Safety Mechanism 10-12
10.4.3 SMU Integration Guidelines 10-13
10.4.4 Alarm Mapping 10-14
10.4.4.1 Pre-alarm Definition 10-14
10.4.4.2 Pre-alarm Signals 10-14
10.4.4.3 Non-compliant Alarms 10-22
10.4.4.4 Internal SMU Alarms 10-23
10.4.4.5 Alarm Signals 10-24
10.4.5 Alarm Handling 10-41
10.4.5.1 Alarm protocol 10-41
10.4.5.2 Alarm Configuration 10-41
10.4.5.3 Alarm operation 10-43
10.4.5.4 Alarm Status Registers 10-44
10.4.5.5 Alarm Debug Registers 10-44
10.4.5.6 Port Emergency Stop 10-45
10.4.5.7 Recovery Timer 10-45
10.4.5.8 Watchdog Alarms 10-46
10.4.6 SMU Control Interface 10-48
10.4.7 SMU state machine 10-50
10.4.8 Fault Signaling Protocol (FSP) 10-52
10.4.8.1 Introduction 10-52
10.4.8.2 Bi-stable fault signaling protocol 10-54
10.4.8.3 Time switching protocol 10-55
10.4.8.4 FSP Fault State 10-56
10.4.8.5 FSP and SMU START State 10-57
10.4.9 OCDS Trigger Bus (OTGB) Interface 10-58
10.4.10 Register Properties 10-59
10.4.10.1 Register Write Protection 10-59
10.4.10.2 Safety Flip-Flops 10-60
10.5 SMU Module Registers 10-62
10.5.1 System Registers description 10-68
10.5.2 SMU Configuration Registers 10-79
10.5.3 SMU Alarm Configuration Registers 10-97
10.5.4 SMU Alarm Configuration Registers (Fault Signaling Protocol) . . 10-108
10.5.5 SMU Alarm Status Registers 10-115
10.5.6 SMU Alarm Debug Registers 10-116
10.5.7 SMU Special Safety Registers: Register Monitor 10-117

Table of Contents

11 Program Memory Unit (PMU) . . . 11-1
11.1 Generic Feature List . . . 11-1
11.2 PMU Configuration of TC21x/TC22x/TC23x . . . 11-2
11.2.1 Features of the BootROM . . . 11-4
11.2.2 Features of the Program and Data Flash . . . 11-4
11.2.2.1 Program Flash Features: . . . 11-4
11.2.2.2 Data Flash Features . . . 11-5
11.3 BootROM . . . 11-6
11.4 Tuning Protection . . . 11-6
11.5 Flash . . . 11-7
11.5.1 Definition of Terms . . . 11-7
11.5.2 Flash Structure . . . 11-8
11.5.2.1 PFlash . . . 11-8
11.5.2.2 DFlash of PMU0 . . . 11-10
11.5.3 Flash Read Access . . . 11-11
11.5.3.1 Read Ports . . . 11-12
11.5.3.2 DFlash, BootROM Read Port . . . 11-13
11.5.3.3 Configuring Flash Wait Cycles . . . 11-13
11.5.3.4 Requested DFlash Read . . . 11-14
11.5.4 Flash Operations . . . 11-14
11.5.4.1 Modes of Operation . . . 11-15
11.5.4.2 Command Sequences . . . 11-15
11.5.4.3 HSM Command Interface . . . 11-16
11.5.4.4 Command Sequence Definitions . . . 11-16
11.5.4.5 Operation Suspend and Resume . . . 11-23
11.5.4.6 Programming Voltage Selection . . . 11-24
11.5.5 Protection . . . 11-24
11.5.5.1 Master Specific Access Control . . . 11-26
11.5.5.2 Register Access Control . . . 11-26
11.5.5.3 Effective Flash Read Protection . . . 11-26
11.5.5.4 Effective Flash Write Protection . . . 11-27
11.5.5.5 Configuring Protection in the UCB . . . 11-28
11.5.5.6 System Wide Effects of Flash Protection . . . 11-36
11.5.6 Data Integrity and Safety . . . 11-38
11.5.6.1 SRI ECC (Safe Fetch Path) . . . 11-38
11.5.6.2 Flash ECC . . . 11-38
11.5.6.3 Margin Checks . . . 11-41
11.5.6.4 PMU and Flash Register Supervision . . . 11-41
11.5.7 Interrupts and Traps . . . 11-41
11.5.8 Reset and Startup . . . 11-42
11.5.9 Power Reduction . . . 11-43
11.6 Signaling to the Safety Management Unit (SMU) . . . 11-43
11.7 Register Set . . . 11-45

11.7.1 PMU Registers . . . 11-45
11.7.1.1 PMU Identification . . . 11-46
11.7.2 Flash Registers . . . 11-48
11.7.2.1 Master Specific Access Control . . . 11-51
11.7.2.2 Flash Identification Register . . . 11-52
11.7.2.3 Flash Status . . . 11-53
11.7.2.4 Flash Configuration Control . . . 11-60
11.7.2.5 Flash Protection . . . 11-63
11.7.2.6 Protection Configuration . . . 11-67
11.7.2.7 Flash Read Buffer Configuration . . . 11-80
11.7.2.8 Requested Read Interface . . . 11-80
11.7.2.9 Flash ECC Access . . . 11-83
11.7.2.10 HSM Command Interface . . . 11-86
11.7.2.11 HSM Requested Read Interface . . . 11-91
11.7.2.12 Margin Check Control . . . 11-95
11.7.2.13 Corrected Bits Address Buffer (CBAB) . . . 11-98
11.7.2.14 Uncorrectable Bits Address Buffer (UBAB) . . . 11-101
11.7.2.15 Direct Flash Communication . . . 11-103
11.8 Application Hints . . . 11-106
11.8.1 Changes with Respect to Audo Families Audo-NG/F/S/Max . . . 11-106
11.8.2 Performing Flash Operations . . . 11-107
11.8.3 EEPROM Emulation With DFlash . . . 11-109
11.8.3.1 Robust EEPROM Emulation . . . 11-110
11.8.4 Handling Errors . . . 11-111
11.8.4.1 Handling Errors During Operation . . . 11-111
11.8.4.2 Handling Errors During Startup . . . 11-115
11.8.5 Resets During Flash Operation . . . 11-115
11.8.5.1 General Advice . . . 11-116
11.8.5.2 Advice for EEPROM Emulation . . . 11-116
11.8.6 ECC . . . 11-117
11.8.7 Startup Tests of ECC Logic . . . 11-118
11.8.7.1 Testing ECC Alarms and Error Flags . . . 11-118
11.8.7.2 Testing the “ECC Monitor” . . . 11-119
11.8.7.3 Testing the SMU Alarm of the “ECC Monitor” . . . 11-120
11.8.7.4 Testing the “EDC Comparator” . . . 11-120
11.8.7.5 General Advice for Startup Tests . . . 11-120

12 Local Memory Unit (LMU) . . . 12-1
12.1 Feature List . . . 12-1
12.2 Local Memory (LMU SRAM) . . . 12-2
12.2.1 LMU SRAM Read Buffers . . . 12-2
12.3 Memory Protection . . . 12-3
12.4 FFT Accelerator Interface . . . 12-4

12.5 Emulation Memory (EMEM) 12-4
12.5.1 EMEM Memory Read Buffers 12-5
12.5.2 Access to Emulation Device Register Space 12-6
12.6 Error Detection and Signalling 12-6
12.6.1 EMEM Read Error 12-6
12.6.2 EMEM Write Error 12-6
12.6.3 Internal ECC Error 12-6
12.6.4 Internal SRAM Read Error 12-6
12.6.5 ECC check failure 12-7
12.6.6 SRI write access data phase error 12-7
12.6.7 SRI access address phase error 12-7
12.7 Online Data Acquisition (OLDA) and its Overlay 12-7
12.8 Clock Control 12-8
12.9 LMU Register Protection 12-8
12.10 LMU Registers 12-9

13 Data Access Overlay (OVC) 13-1
13.1 Data Access Redirection 13-2
13.2 Target Memories 13-4
13.2.1 Online Data Acquisition (OLDA) Space 13-4
13.3 Overlay Memories 13-4
13.3.1 Local Memory 13-4
13.3.2 Emulation Memory 13-5
13.3.3 DSPR & PSPR Memory 13-5
13.4 Global Overlay Control 13-5
13.4.1 Global Overlay Control Synchronisation 13-6
13.5 Overlay Configuration Change 13-7
13.6 Access Protection, Attributes, Concurrent Matches 13-7
13.7 Overlay Control Registers 13-8
13.7.1 Block control registers 13-9
13.8 Global overlay control registers 13-15

14 General Purpose I/O Ports and Peripheral I/O Lines (Ports) 14-1
14.1 Basic Port Operation 14-1
14.2 Description Scheme for the Port IO Functions 14-5
14.3 Port Register Description 14-7
14.3.1 Module Identification Register 14-12
14.3.2 Port Input/Output Control Registers 14-13
14.3.3 Pad Driver Mode Register 14-19
14.3.4 Pin Function Decision Control Register 14-22
14.3.5 Pin Controller Select Register 14-23
14.3.6 Port Output Register 14-24
14.3.7 Port Output Modification Register 14-26
14.3.8 Port Output Modification Set Register 14-28

14.3.9 Port Output Modification Set Registers x 14-29
14.3.10 Port Output Modification Clear Register 14-33
14.3.11 Port Output Modification Clear Registers x 14-34
14.3.12 Emergency Stop Register 14-38
14.3.13 Port Input Register 14-39
14.3.14 Access Protection Registers 14-41
14.4 Port 00 14-44
14.4.1 Port 00 Registers 14-45
14.4.1.1 Port 00 Output Register 14-46
14.4.1.2 Port 00 Output Modification Register 14-46
14.4.1.3 Port 00 Output Modification Set Register 14-46
14.4.1.4 Port 00 Output Modification Set Register 12 14-46
14.4.1.5 Port 00 Output Modification Clear Register 14-46
14.4.1.6 Port 00 Output Modification Clear Register 12 14-46
14.4.1.7 Port 00 Input/Output Control Register 12 14-47
14.4.1.8 Port 00 Input Register 14-47
14.4.1.9 Port 00 Pad Driver Mode 1 Register 14-48
14.4.1.10 Port 00 Emergency Stop Register 14-48
14.5 Port 02 14-49
14.5.1 Port 02 Registers 14-50
14.5.1.1 Port 02 Output Register 14-51
14.5.1.2 Port 02 Output Modification Register 14-51
14.5.1.3 Port 02 Output Modification Set Register 14-51
14.5.1.4 Port 02 Output Modification Set Register 8 14-51
14.5.1.5 Port 02 Output Modification Clear Register 14-51
14.5.1.6 Port 02 Output Modification Clear Register 8 14-51
14.5.1.7 Port 02 Input/Output Control Register 8 14-52
14.5.1.8 Port 02 Input Register 14-52
14.5.1.9 P02 Pad Driver Mode 1 Register 14-53
14.5.1.10 Port 02 Emergency Stop Register 14-53
14.6 Port 10 14-54
14.6.1 Port 10 Registers 14-55
14.6.1.1 Port 10 Output Register 14-55
14.6.1.2 Port 10 Output Modification Register 14-56
14.6.1.3 Port 10 Output Modification Set Register 14-56
14.6.1.4 Port 10 Output Modification Set Register 0 14-56
14.6.1.5 Port 10 Output Modification Set Register 4 14-56
14.6.1.6 Port 10 Output Modification Clear Register 14-56
14.6.1.7 Port 10 Output Modification Clear Register 0 14-56
14.6.1.8 Port 10 Output Modification Clear Register 4 14-56
14.6.1.9 Port 10 Input/Output Control Register 0 14-57
14.6.1.10 Port 10 Input/Output Control Register 4 14-58
14.6.1.11 Port 10 Input Register 14-58

14.6.1.12 Port 10 Pad Driver Mode 0 Register 14-59
14.6.1.13 Port 10 Emergency Stop Register 14-59
14.7 Port 11 14-60
14.7.1 Port 11 Registers 14-61
14.7.1.1 Port 11 Output Register 14-62
14.7.1.2 Port 11 Output Modification Register 14-62
14.7.1.3 Port 11 Output Modification Set Register 14-62
14.7.1.4 Port 11 Output Modification Set Register 0 14-62
14.7.1.5 Port 11 Output Modification Set Register 4 14-62
14.7.1.6 Port 11 Output Modification Set Register 12 14-62
14.7.1.7 Port 11 Output Modification Clear Register 14-62
14.7.1.8 Port 11 Output Modification Clear Register 0 14-63
14.7.1.9 Port 11 Output Modification Clear Register 4 14-63
14.7.1.10 Port 11 Output Modification Clear Register 12 14-63
14.7.1.11 Port 11 Input/Output Control Register 0 14-64
14.7.1.12 Port 11 Input/Output Control Register 4 14-65
14.7.1.13 Port 11 Input/Output Control Register 12 14-66
14.7.1.14 Port 11 Input Register 14-66
14.7.1.15 Port 11 Pad Driver Mode 0 Register 14-67
14.7.1.16 Port 11 Pad Driver Mode 1 Register 14-68
14.7.1.17 Port 11 Emergency Stop Register 14-68
14.8 Port 13 14-69
14.8.1 Port 13 Registers 14-70
14.8.1.1 Port 13 Output Register 14-70
14.8.1.2 Port 13 Output Modification Register 14-70
14.8.1.3 Port 13 Output Modification Set Register 14-71
14.8.1.4 Port 13 Output Modification Clear Register 14-71
14.8.1.5 Port 13 Input Register 14-71
14.8.1.6 Port 13 Pad Driver Mode 0 Register 14-72
14.8.1.7 Port 13 Emergency Stop Register 14-72
14.9 Port 14 14-73
14.9.1 Port 14 Registers 14-74
14.9.1.1 Port 14 Output Register 14-75
14.9.1.2 Port 14 Output Modification Register 14-75
14.9.1.3 Port 14 Output Modification Set Register 14-75
14.9.1.4 Port 14 Output Modification Set Register 8 14-75
14.9.1.5 Port 14 Output Modification Clear Register 14-75
14.9.1.6 Port 14 Output Modification Clear Register 8 14-75
14.9.1.7 Port 14 Input/Output Control Register 8 14-76
14.9.1.8 Port 14 Input Register 14-76
14.9.1.9 P14 Pad Driver Mode 1 Register 14-77
14.9.1.10 Port 14 Emergency Stop Register 14-77
14.10 Port 15 14-78

14.10.1 Port 15 Registers 14-79
14.10.1.1 Port 15 Output Register 14-80
14.10.1.2 Port 15 Output Modification Register 14-80
14.10.1.3 Port 15 Output Modification Set Register 14-80
14.10.1.4 Port 15 Output Modification Set Register 8 14-80
14.10.1.5 Port 15 Output Modification Clear Register 14-80
14.10.1.6 Port 14 Output Modification Clear Register 8 14-80
14.10.1.7 Port 14 Input/Output Control Register 8 14-81
14.10.1.8 Port 15 Input Register 14-81
14.10.1.9 P15 Pad Driver Mode 1 Register 14-82
14.10.1.10 Port 15 Emergency Stop Register 14-82
14.11 Port 20 14-83
14.11.1 Port 20 Registers 14-84
14.11.1.1 Port 20 Output Register 14-85
14.11.1.2 Port 20 Output Modification Register 14-85
14.11.1.3 Port 20 Output Modification Set Register 14-85
14.11.1.4 Port 20 Output Modification Set Register 0 14-85
14.11.1.5 Port 20 Output Modification Set Register 4 14-85
14.11.1.6 Port 20 Output Modification Set Register 12 14-85
14.11.1.7 Port 20 Output Modification Clear Register 14-86
14.11.1.8 Port 20 Output Modification Clear Register 0 14-86
14.11.1.9 Port 20 Output Modification Clear Register 4 14-86
14.11.1.10 Port 20 Output Modification Clear Register 12 14-86
14.11.1.11 Port 20 Input/Output Control Register 0 14-87
14.11.1.12 Port 20 Input/Output Control Register 4 14-88
14.11.1.13 Port 20 Input/Output Control Register 12 14-89
14.11.1.14 Port 20 Input Register 14-89
14.11.1.15 Port 20 Pad Driver Mode 0 Register 14-90
14.11.1.16 Port 20 Pad Driver Mode 1 Register 14-91
14.11.1.17 Port 20 Emergency Stop Register 14-92
14.12 Port 21 14-93
14.12.1 Port 21 Registers 14-94
14.12.1.1 Port 21 Output Register 14-94
14.12.1.2 Port 21 Output Modification Register 14-95
14.12.1.3 Port 21 Output Modification Set Register 14-95
14.12.1.4 Port 21 Output Modification Set Register 0 14-95
14.12.1.5 Port 21 Output Modification Clear Register 14-95
14.12.1.6 Port 21 Output Modification Clear Register 0 14-95
14.12.1.7 Port 21 Input/Output Control Register 0 14-96
14.12.1.8 Port 21 Pad Driver Mode 0 Register 14-97
14.12.1.9 Port 21 Input Register 14-98
14.12.1.10 Port 21 Emergency Stop Register 14-98
14.13 Port 22 14-99

14.13.1 Port 22 Registers 14-100
14.13.1.1 Port 22 Output Register 14-101
14.13.1.2 Port 22 Output Modification Register 14-101
14.13.1.3 Port 22 Output Modification Set Register 14-101
14.13.1.4 Port 22 Output Modification Set Register 4 14-101
14.13.1.5 Port 22 Output Modification Clear Register 14-101
14.13.1.6 Port 22 Output Modification Clear Register 4 14-101
14.13.1.7 Port 22 Input Register 14-101
14.13.1.8 Port 22 Input/Output Control Register 4 14-102
14.13.1.9 Port 22 Pad Driver Mode 0 Register 14-103
14.13.1.10 Port 22 Emergency Stop Register 14-103
14.14 Port 23 14-104
14.14.1 Port 23 Configuration 14-104
14.14.2 Port 23 Registers 14-105
14.14.2.1 Port 23 Output Register 14-105
14.14.2.2 Port 23 Input/Output Control Register 0 14-107
14.14.2.3 Port 23 Pad Driver Mode 0 Register 14-108
14.14.2.4 Port 23 Output Modification Register 14-108
14.14.2.5 Port 23 Output Modification Set Register 14-109
14.14.2.6 Port 23 Output Modification Set Register 0 14-109
14.14.2.7 Port 23 Output Modification Clear Register 14-109
14.14.2.8 Port 23 Output Modification Clear Register 0 14-109
14.14.2.9 Port 23 Input Register 14-109
14.14.2.10 Port 23 Emergency Stop Register 14-109
14.15 Port 33 14-110
14.15.1 Port 33 Registers 14-111
14.15.1.1 Port 33 Output Register 14-112
14.15.1.2 Port 33 Output Modification Register 14-112
14.15.1.3 Port 33 Output Modification Set Register 14-112
14.15.1.4 Port 33 Output Modification Set Register 12 14-112
14.15.1.5 Port 33 Output Modification Clear Register 14-112
14.15.1.6 Port 33 Output Modification Clear Register 12 14-112
14.15.1.7 Port 33 Input/Output Control Register 12 14-113
14.15.1.8 Port 33 Input Register 14-113
14.15.1.9 Port 33 Pad Driver Mode 1 Register 14-114
14.15.1.10 Port 33 Emergency Stop Register 14-114
14.16 Port 34 14-115
14.16.1 Port 34 Registers 14-116
14.16.1.1 Port 34 Output Register 14-116
14.16.1.2 Port 34 Output Modification Register 14-116
14.16.1.3 Port 34 Output Modification Set Register 14-117
14.16.1.4 Port 34 Output Modification Clear Register 14-117
14.16.1.5 Port 34 Input Register 14-117

14.16.1.6 Port 34 Pad Driver Mode 0 Register 14-118
14.16.1.7 Port 34 Emergency Stop Register 14-118
14.17 Port 40 14-119
14.17.1 Port 40 Configuration 14-119
14.17.2 Port 40 Registers 14-120
14.17.3 Port 40 Input/Output Control Registers 14-121
14.17.4 Port 40 Pin Function Decision Control Register 14-123
14.17.5 Port 40 Pin Controller Select Register 14-126
14.18 Port 41 14-128
14.18.1 Port 41 Configuration 14-128
14.18.2 Port 41 Registers 14-129
14.18.3 Port 41 Input/Output Control Registers 14-130
14.18.4 Port 41 Pin Function Decision Control Register 14-132
14.18.5 Port 41 Pin Controller Select Register 14-135
14.19 Register Behavior for Unavailable Port Pins 14-137

15 Direct Memory Access (DMA) 15-1
15.1 What is new 15-1
15.2 Features 15-2
15.3 Block Diagram 15-4
15.4 Functional Description 15-6
15.4.1 Definition of Terms 15-6
15.4.2 DMA Principles 15-7
15.4.3 DMA Channel Functionality 15-8
15.4.3.1 Shadowed Source or Destination Address 15-8
15.4.3.2 DMA Channel Request Control 15-13
15.4.3.3 DMA Channel Operation Modes 15-14
15.4.3.4 DMA Service Requests 15-19
15.4.3.5 Channel Reset Operation 15-20
15.4.3.6 Channel Halt Operation 15-21
15.4.3.7 Transfer Count and Move Count 15-23
15.4.3.8 Circular Buffer 15-25
15.4.3.9 Address Counter 15-26
15.4.3.10 Flow Control 15-26
15.4.3.11 Double Buffering Operation 15-29
15.4.3.12 Linked Lists 15-37
15.4.3.13 DMA Linked List 15-38
15.4.3.14 Accumulated Linked List 15-40
15.4.3.15 Safe Linked List 15-40
15.4.3.16 Conditional Linked List 15-42
15.4.4 Transaction Control Engine 15-45
15.4.4.1 Error Conditions 15-47
15.4.5 Bus Switch, Bus Switch Priorities 15-48

15.4.6 DMA Module Priorities on On Chip Busses 15-51
15.4.6.1 On Chip Bus Access Rights, RMW support 15-51
15.4.6.2 On Chip Bus Master Interfaces 15-51
15.4.7 Pattern Detection 15-53
15.4.7.1 Pattern Compare Logic 15-53
15.4.7.2 Pattern Detection for 8-bit Data Width 15-54
15.4.7.3 Pattern Detection for 16-bit Data Width 15-56
15.4.7.4 Pattern Detection for 32-bit Data Width 15-59
15.4.8 DMA Configuration Interface 15-61
15.4.8.1 DMARAM Channel Control and Status Word 15-61
15.4.8.2 DMA Active Channel Write Back 15-61
15.4.8.3 DMA Active Channel Shadow Control 15-62
15.4.8.4 DMARAM Write Back During Linked List Execution 15-63
15.4.9 Interrupt Service Requests 15-64
15.4.9.1 Channel Transfer Interrupt Service Request 15-66
15.4.9.2 Channel Pattern Detection Interrupt Service Request 15-66
15.4.9.3 Channel Wrap Buffer Interrupt Service Request 15-68
15.4.9.4 Transaction Request Lost Interrupt Service Request 15-69
15.4.9.5 Source and Destination Error Interrupt Service Requests 15-70
15.4.9.6 DMA Linked List Error Interrupt Service Request 15-71
15.5 Power Modes 15-72
15.5.1 Sleep Mode 15-72
15.6 Functional Safety Features 15-72
15.6.1 Access Protection 15-72
15.6.2 Data Integrity 15-73
15.6.2.1 DMARAM 15-73
15.6.2.2 DMA SRI Read and Write Data 15-74
15.7 Debug Features 15-74
15.7.1 Channel Suspend Mode 15-74
15.7.2 OCDS Trigger Bus (OTGB) Interface 15-75
15.7.3 MCDS Trace Interface 15-76
15.8 Register Description 15-77
15.8.1 DMA General Module Control Registers 15-86
15.8.2 DMA Access Protection Registers 15-90
15.8.3 DMA Sub-block Error Registers 15-92
15.8.4 DMA Sub-block Move Engine Registers 15-99
15.8.5 DMA Move Engine Active Channel Registers 15-108
15.8.6 DMA OCDS Registers 15-130
15.8.7 DMA Pattern Detection Registers 15-134
15.8.8 DMA Flow Control Registers 15-136
15.8.9 DMA Channel Hardware Resource Registers 15-137
15.8.10 DMA Channel Suspend Registers 15-139
15.8.11 DMA Transaction State Registers 15-141

15.8.12 DMA Transaction Control Set 15-144
15.9 Use Cases 15-156

16 Interrupt Router (IR) 16-1
16.1 Overview 16-1
16.2 Features 16-2
16.3 Service Request Nodes (SRN) 16-3
16.3.1 Service Request Control Registers 16-3
16.3.1.1 General Service Request Control Register Format 16-3
16.3.1.2 Changing the SRN configuration 16-7
16.3.1.3 Protection of the SRC Registers 16-7
16.3.1.4 Request Set and Clear Bits (SETR, CLRR) 16-8
16.3.1.5 Enable Bit (SRE) 16-8
16.3.1.6 Service Request Flag (SRR) 16-8
16.3.1.7 Type-Of-Service Control (TOS) 16-9
16.3.1.8 Service Request Priority Number (SRPN) 16-9
16.3.1.9 ECC Encoding (ECC) 16-10
16.3.1.10 Interrupt Trigger Overflow Bit (IOV) 16-11
16.3.1.11 Interrupt Trigger Overflow Clear Bit (IOVCLR) 16-11
16.3.1.12 SW Sticky Bit (SWS) 16-11
16.3.1.13 SW Sticky Clear Bit (SWSCLR) 16-11
16.4 Interrupt Control Unit (ICU) 16-12
16.4.1 ICU Control Registers 16-12
16.4.1.1 Latest Winning Service Request Register (LWSR) 16-13
16.4.1.2 Last Acknowledged Service Request Register (LASR) 16-15
16.4.1.3 Error Capture Register (ECR) 16-16
16.5 General Purpose Service Requests, Service Request Broadcast 16-17
16.5.1 General Purpose Service Requests (GPSRxy) 16-18
16.5.2 Service Request Broadcast Registers (SRBx) 16-18
16.6 System Registers 16-19
16.6.1 Register Access Protection (ACCEN1/0) 16-19
16.6.2 Kernel Reset Registers (KRST1/0, KRSTCLR) 16-20
16.6.3 Clock Control Register (CLC) 16-20
16.6.4 OCDS Control and Status Register (OCS) 16-20
16.7 Arbitration Process 16-21
16.7.1 Number of Clock Cycles per Arbitration Process 16-22
16.7.2 Service Request Acknowledge 16-24
16.7.3 Handling of detected ECC Errors 16-24
16.8 Usage of the TC21x/TC22x/TC23x Interrupt System 16-25
16.8.1 CPU to ICU Interface 16-25
16.8.2 DMA to ICU Interface 16-25
16.8.3 Software-Initiated Interrupts 16-25
16.8.4 External Interrupts 16-26

16.9 Use Case Examples 16-26
16.9.1 Use Case Example Interrupt Handler 16-27
16.10 Module Implementation 16-29
16.10.1 Characteristics of TC21x/TC22x/TC23x Interrupt Router Module . . 16-29
16.10.2 Mapping of TC21x/TC22x/TC23x Module Service Request Triggers to SRNs 16-29
16.10.2.1 Mapping of Service Request Control Registers 16-30
16.10.2.2 Interrupts related to the Debug Reset 16-31
16.10.2.3 Timing characteristics of Service Request Trigger Signals 16-31
16.11 Interrupt Router System and Module Registers 16-33
16.11.1 System and ICU Control Registers 16-37
16.12 OTGM Registers 16-43
16.12.1 Status and Control 16-44
16.12.2 IRQ MUX Control 16-45
16.12.3 Interrupt System Trace 16-47
16.12.4 MCDS Interface 16-48
16.13 Interrupt Router SRC Registers 16-50

17 System Timer (STM) 17-1
17.1 Overview 17-1
17.2 Operation 17-1
17.2.1 Compare Register Operation 17-3
17.2.2 Compare Match Interrupt Control 17-4
17.2.3 STM as Reset Trigger 17-5
17.3 STM Registers 17-5
17.3.1 Clock Control Register 17-7
17.3.2 Timer/Capture Registers 17-9
17.3.3 Compare Registers 17-12
17.3.4 Interrupt Registers 17-16
17.3.5 Interface Registers 17-19

18 On-Chip Debug Support (OCDS) 18-1
18.1 Feature List 18-2
18.2 Family Overview 18-4
18.3 Tool Interface Recommendations 18-4
18.4 Debug Access Server (DAS) 18-6

19 Asynchronous/Synchronous Interface (ASCLIN) 19-1
19.1 Feature List 19-2
19.2 Overview 19-4
19.3 External Signals 19-5
19.4 User Interface 19-6
19.4.1 TxFIFO Overview 19-6
19.4.2 Using the TxFIFO 19-7

19.4.2.1 Standard ASC Mode . . . 19-7
19.4.2.2 High Speed ASC Mode . . . 19-9
19.4.2.3 LIN Mode . . . 19-11
19.4.2.4 SPI Mode . . . 19-11
19.4.3 RxFIFO Overview . . . 19-13
19.4.4 Using the RxFIFO . . . 19-15
19.4.4.1 Standard ASC Mode . . . 19-15
19.4.4.2 High Speed ASC Mode . . . 19-17
19.4.4.3 LIN Mode . . . 19-17
19.4.4.4 SPI Mode . . . 19-19
19.4.5 RTS / CTS Handshaking . . . 19-19
19.5 Clock System . . . 19-20
19.5.1 Baud Rate Generation . . . 19-20
19.5.2 Bit Timing Properties . . . 19-21
19.6 Data Frame Configuration . . . 19-24
19.7 Miscellaneous Configuration . . . 19-24
19.8 Synchronous Mode . . . 19-25
19.8.1 Baud Rate and Clock Generation . . . 19-25
19.8.2 Data Frame Configuration . . . 19-26
19.8.3 Slave Selects Configuration . . . 19-26
19.8.4 Miscellaneous Configuration . . . 19-26
19.9 LIN Support . . . 19-27
19.9.1 LIN Watchdog . . . 19-29
19.9.1.1 LIN Break, Wake, Stuck Handling . . . 19-30
19.9.1.2 LIN Header and Response Timers . . . 19-32
19.9.2 LIN Master Sequences . . . 19-34
19.9.3 LIN Slave Sequences . . . 19-38
19.9.4 Using the ENI and HO Bits . . . 19-39
19.9.5 LIN Error Recovery . . . 19-40
19.9.6 LIN Sleep and LIN Wake-Up . . . 19-41
19.10 Auto Baud Rate Detection . . . 19-42
19.11 Collision Detection . . . 19-44
19.12 LIN Protocol Control . . . 19-45
19.13 Interrupts . . . 19-47
19.14 Digital Glitch Filter . . . 19-50
19.15 Suspend, Sleep and Power-Off Behavior . . . 19-51
19.15.1 OCDS Suspend . . . 19-51
19.15.2 Sleep Mode . . . 19-51
19.15.3 Disable Request (Power-Off) . . . 19-52
19.16 Reset Behavior . . . 19-52
19.17 Use Case Example ASC Interface . . . 19-53
19.18 Kernel Registers . . . 19-56
19.18.1 Kernel Registers . . . 19-60

19.19 Implementation 19-106
19.19.1 BPI_FPI Module Registers 19-106
19.19.1.1 System Registers 19-106
19.20 On-Chip Connections 19-114
19.21 ASC at CAN Support 19-117

20 Queued Synchronous Peripheral Interface (QSPI) 20-1
20.1 Feature List 20-1
20.2 Overview 20-3
20.2.1 External Signals 20-3
20.2.2 Operating Modes 20-4
20.2.3 Queue Support Overview 20-6
20.2.4 Architecture Overview 20-7
20.2.5 Three Wire Connection 20-8
20.3 Abstract Overview 20-9
20.4 Frequency Domains 20-10
20.5 Master Mode State Machine 20-11
20.5.1 Phases of one Communication Cycle 20-11
20.5.2 Configuration Extensions 20-17
20.5.3 Details of the Baud Rate and Phase Duration Control 20-18
20.5.4 Calculation of the Baud Rates and the Delays 20-20
20.5.5 State Diagram of Standard Communication Cycle 20-21
20.5.6 Expect Phase 20-24
20.5.7 External Slave Select Expansion 20-25
20.6 User Interface 20-26
20.6.1 Transmit and Receive FIFOs 20-27
20.6.1.1 Short Data Mode 20-30
20.6.1.2 Long Data Mode 20-32
20.6.1.3 Continuous Data Mode 20-35
20.6.1.4 Single Configuration - Multiple Frames Behavior 20-38
20.6.1.5 Big Endian Data Format 20-38
20.6.2 Loop-Back Mode 20-40
20.7 Interrupts 20-42
20.7.1 Slave Mode SLSI Interrupt 20-43
20.7.2 Interrupt Flags Behavior 20-44
20.7.3 TXFIFO Interrupt Generation 20-45
20.7.4 RXFIFO Interrupt Generation 20-49
20.7.5 DMA Transfer Example 20-52
20.8 Slave Mode 20-53
20.8.1 Shift Clock Phase and Polarity in Slave Mode 20-54
20.8.2 Shift Clock Monitoring 20-55
20.8.2.1 Baud Rate Error Detection 20-56
20.8.2.2 Spike Detection 20-56

20.8.2.3 Shift Clock Monitor Flags 20-57
20.8.3 Parity 20-58
20.9 Kernel Registers 20-59
20.9.1 Kernel Registers 20-62
20.10 Operation Modes 20-94
20.10.1 OCDS Suspend 20-97
20.10.2 Sleep Mode 20-99
20.10.3 Disabling the QSPI 20-100
20.11 Reset Behavior 20-101
20.12 QSPI Module Implementation 20-102
20.12.1 Module Identification Registers 20-102
20.12.2 Interfaces of the QSPI Modules 20-102
20.12.3 On-Chip Connections 20-103
20.12.4 QSPI Related External Registers 20-104
20.12.4.1 BPI_FPI Module Registers 20-105
20.12.4.2 Interrupt Control Registers 20-113
20.13 High Speed Input Capture 20-114
20.13.1 HSIC Registers 20-117
20.13.1.1 Register Description 20-117
20.13.2 HSIC Implementation 20-118
20.13.2.1 Connection to the pins 20-119

21 Controller Area Network Controller (MultiCAN+) 21-1
21.1 CAN Basics 21-2
21.1.1 Addressing and Bus Arbitration 21-2
21.1.2 CAN Frame Formats 21-3
21.1.3 CAN Frame Types 21-3
21.1.3.1 Data Frames 21-3
21.1.3.2 Remote Frames 21-7
21.1.3.3 Error Frames 21-7
21.1.3.4 Overload Frame 21-8
21.1.4 The Nominal Bit Time 21-9
21.1.4.1 CAN FD bit timing 21-9
21.1.5 Error Detection and Error Handling 21-10
21.2 Overview 21-12
21.2.1 Features List 21-13
21.3 CAN Flexible Data-Rate (CAN FD) 21-16
21.3.1 Transmitter Delay Compensation 21-16
21.4 MultiCAN+ Kernel Functional Description 21-18
21.4.1 Module Structure 21-18
21.4.2 Clock Control 21-20
21.4.3 Port Input Control 21-25
21.4.4 OCDS Suspend 21-25

21.4.5 OCDS Trigger Bus (OTGB) Interface 21-26
21.4.6 CAN Node Control 21-27
21.4.6.1 Bit Timing Unit 21-28
21.4.6.2 Bitstream Processor 21-30
21.4.6.3 Error Handling Unit 21-30
21.4.6.4 CAN Frame Counter 21-31
21.4.6.5 Node Timing Functions 21-32
21.4.6.6 CAN Node Interrupts 21-34
21.4.7 Message Object List Structure 21-36
21.4.7.1 Basics 21-36
21.4.7.2 List of Unallocated Elements 21-37
21.4.7.3 Connection to the CAN Nodes 21-37
21.4.7.4 List Command Panel 21-38
21.4.8 CAN Node Analyzer Mode 21-40
21.4.8.1 Analyzer Mode 21-40
21.4.8.2 Loop-Back Mode 21-41
21.4.8.3 Bit Timing Analysis 21-42
21.4.9 Message Acceptance Filtering 21-43
21.4.9.1 Receive Acceptance Filtering 21-43
21.4.9.2 Transmit Acceptance Filtering 21-45
21.4.10 Message Postprocessing 21-46
21.4.10.1 Message Object Interrupts 21-46
21.4.10.2 Pending Messages 21-48
21.4.11 Message Object Data Handling 21-50
21.4.11.1 Frame Reception 21-50
21.4.11.2 Frame Transmission 21-53
21.4.12 Message Object Functionality 21-56
21.4.12.1 Standard Message Object 21-56
21.4.12.2 Single Data Transfer Mode 21-56
21.4.12.3 Single Transmit Trial 21-56
21.4.12.4 Message Object Format (Classical CAN & CAN FD) 21-57
21.4.12.5 Message Object FIFO Structure 21-57
21.4.12.6 Receive FIFO 21-60
21.4.12.7 Transmit FIFO 21-60
21.4.12.8 Gateway Mode 21-61
21.4.12.9 Foreign Remote Requests 21-64
21.4.12.10 CAN FD - 64 byte Messages 21-64
21.4.13 Measurement for Oscillator Calibration 21-65
21.4.14 Debug Over CAN 21-66
21.5 Use Case Example MultiCAN+ 21-67
21.6 MultiCAN+ Kernel Registers 21-72
21.6.1 Global Module Registers 21-78
21.6.2 CAN Node Registers 21-95

21.6.3 Message Object Registers . . . 21-125
21.7 MultiCAN+ Module Implementation . . . 21-152
21.7.1 Interfaces of the MultiCAN+ Module . . . 21-152
21.7.2 MultiCAN+ Module External Registers . . . 21-153
21.7.2.1 System Registers . . . 21-155
21.7.3 MultiCAN+ Clock Interconnects With SCU . . . 21-163
21.7.4 Module Clock Generation . . . 21-165
21.7.4.1 Clock Selection . . . 21-165
21.7.4.2 Fractional Divider . . . 21-165
21.7.5 Port and I/O Line Control . . . 21-169
21.7.5.1 Input/Output Function Selection in Ports . . . 21-169
21.7.5.2 Node Receive Input Selection . . . 21-170
21.7.5.3 CAN Transmit Trigger Inputs . . . 21-171
21.7.5.4 Connections to Interrupt Router Inputs . . . 21-172
21.7.5.5 Connections to General Timer Module (GTM) Inputs . . . 21-173
21.7.6 Interrupt Control . . . 21-174
21.7.7 MultiCAN+ Module Register Address Map . . . 21-176
21.8 Revision history . . . 21-177

22 Single Edge Nibble Transmission (SENT) . . . 22-1
22.1 SENT Kernel Description . . . 22-1
22.1.1 Overview . . . 22-3
22.1.2 General Operation . . . 22-5
22.1.2.1 Definitions . . . 22-6
22.1.3 Standard SENT Operation . . . 22-7
22.1.3.1 Frame Formats and Definitions . . . 22-8
22.1.4 SPC Operation . . . 22-14
22.1.4.1 Synchronous Transmission . . . 22-14
22.1.4.2 Range Selection . . . 22-14
22.1.4.3 ID Selection . . . 22-15
22.1.4.4 Bidirectional Transmit Mode . . . 22-16
22.1.4.5 SPC Timing . . . 22-16
22.1.4.6 Abort of Frames . . . 22-17
22.1.5 Baud Rate Generation . . . 22-18
22.1.6 Error Detection Capabilities . . . 22-20
22.1.7 Digital Glitch Filter. . . . 22-21
22.1.8 Interrupts . . . 22-22
22.1.9 Trigger Outputs . . . 22-22
22.2 SENT Kernel Registers . . . 22-23
22.2.1 Module Control . . . 22-27
22.2.2 Channel Baud Rate Registers . . . 22-31
22.2.3 Receiver Control and Status Registers . . . 22-33
22.2.4 Input and Output Control . . . 22-43

22.2.5 Receive Data Registers . . . 22-47
22.2.6 SPC Control . . . 22-51
22.2.7 Interrupt Control Registers . . . 22-54
22.3 SENT Module Implementation . . . 22-71
22.3.1 Interface Connections of the SENT Module . . . 22-71
22.3.1.1 On-Chip Connections . . . 22-72
22.3.1.2 Interrupt and DMA Controller Service Requests . . . 22-72
22.3.2 SENT Module-Related External Registers . . . 22-74
22.3.2.1 Port Control . . . 22-74
22.3.3 BPI_FPI Module Registers . . . 22-77
22.4 Revision History . . . 22-84

23 FlexRay™ Protocol Controller (E-Ray) . . . 23-1
23.1 E-Ray Kernel Description . . . 23-1
23.2 Overview . . . 23-2
23.3 Definitions . . . 23-3
23.4 Block Diagram . . . 23-3
23.5 Programmer's Model . . . 23-6
23.5.1 Register Map . . . 23-6
23.5.2 E-Ray Kernel Registers . . . 23-8
23.5.2.1 Customer Registers . . . 23-15
23.5.2.2 Special Registers . . . 23-22
23.5.2.3 Service Request Registers . . . 23-32
23.5.2.4 Communication Controller Control Registers . . . 23-79
23.5.2.5 Communication Controller Status Registers . . . 23-106
23.5.2.6 Message Buffer Control Registers . . . 23-129
23.5.2.7 Message Buffer Status Registers . . . 23-136
23.5.2.8 Identification Registers . . . 23-156
23.5.2.9 Input Buffer . . . 23-158
23.5.2.10 Output Buffer . . . 23-169
23.6 Functional Description . . . 23-187
23.6.1 Communication Cycle . . . 23-187
23.6.1.1 Static Segment . . . 23-187
23.6.1.2 Dynamic Segment . . . 23-188
23.6.1.3 Symbol Window . . . 23-188
23.6.1.4 Network Idle Time (NIT) . . . 23-188
23.6.1.5 Configuration of Network Idle Time (NIT) Start and Offset Correction Start . . . 23-188
23.6.2 Communication Modes . . . 23-190
23.6.3 Clock Synchronization . . . 23-190
23.6.3.1 Global Time . . . 23-190
23.6.3.2 Local Time . . . 23-190
23.6.3.3 Synchronization Process . . . 23-191

23.6.3.4 External Clock Synchronization . . . 23-192
23.6.4 Error Handling . . . 23-193
23.6.4.1 Clock Correction Failed Counter . . . 23-193
23.6.4.2 Passive to Active Counter . . . 23-194
23.6.4.3 HALT Command . . . 23-194
23.6.4.4 FREEZE Command . . . 23-194
23.6.5 Communication Controller States . . . 23-196
23.6.5.1 Communication Controller State Diagram . . . 23-196
23.6.5.2 DEFAULT_CONFIG State . . . 23-198
23.6.5.3 MONITOR_MODE . . . 23-199
23.6.5.4 READY State . . . 23-200
23.6.5.5 WAKEUP State . . . 23-200
23.6.5.6 STARTUP State . . . 23-205
23.6.5.7 Startup Time-outs . . . 23-208
23.6.5.8 Path of leading Coldstart Node (initiating coldstart) . . . 23-209
23.6.5.9 NORMAL_ACTIVE State . . . 23-211
23.6.5.10 NORMAL_PASSIVE State . . . 23-211
23.6.5.11 HALT State . . . 23-212
23.6.6 Network Management . . . 23-213
23.6.7 Filtering and Masking . . . 23-213
23.6.7.1 Frame ID Filtering . . . 23-214
23.6.7.2 Channel ID Filtering . . . 23-214
23.6.7.3 Cycle Counter Filtering . . . 23-215
23.6.7.4 FIFO Filtering . . . 23-216
23.6.8 Transmit Process . . . 23-217
23.6.8.1 Static Segment . . . 23-217
23.6.8.2 Dynamic Segment . . . 23-217
23.6.8.3 Transmit Buffers . . . 23-217
23.6.8.4 Frame Transmission . . . 23-218
23.6.8.5 NULL Frame Transmission . . . 23-219
23.6.9 Receive Process . . . 23-220
23.6.9.1 Frame Reception . . . 23-220
23.6.9.2 NULL Frame reception . . . 23-221
23.6.10 FIFO Function . . . 23-221
23.6.10.1 Description . . . 23-221
23.6.10.2 Configuration of the FIFO . . . 23-222
23.6.10.3 Access to the FIFO . . . 23-223
23.6.11 Message Handling . . . 23-223
23.6.11.1 Host access to Message RAM . . . 23-223
23.6.11.2 Data Transfers between IBF / OBF and Message RAM . . . 23-228
23.6.11.3 Minimum f_{CLC_ERAY} . . . 23-234
23.6.11.4 FlexRay™ Protocol Controller access to Message RAM . . . 23-238
23.6.12 Message RAM . . . 23-239

23.6.12.1 Header Partition 23-241
23.6.12.2 Data Partition 23-244
23.6.12.3 ECC Check 23-245
23.6.13 Host Handling of Errors 23-248
23.6.13.1 Self-Healing 23-248
23.6.13.2 CLEAR_RAMS Command 23-248
23.6.13.3 Temporary Unlocking of Header Section 23-248
23.7 Module Service Request 23-250
23.8 Restrictions 23-253
23.8.1 Message Buffers with the same Frame ID 23-253
23.8.2 Data Transfers between IBF / OBF and Message RAM 23-253
23.9 E-Ray Module Implementation 23-254
23.9.1 Interconnections of the E-Ray Module 23-254
23.9.2 Port Control and Connections 23-255
23.9.2.1 Input/Output Function Selection 23-255
23.9.3 On-Chip Connections 23-257
23.9.3.1 E-Ray Connections with IR 23-257
23.9.3.2 E-Ray Connections with SMU 23-257
23.9.3.3 E-Ray Connections with the External Request Unit of SCU . . . 23-257
23.9.3.4 E-Ray Connections to GTM 23-258
23.9.3.5 E-Ray Connections with the External Clock Output of SCU . . . 23-258
23.9.4 OCDS Trigger Bus (OTGB) Interface 23-258
23.9.5 OTGB E-Ray Registers 23-261
23.9.5.1 OCDS Trigger Bus (OTGB) 23-261
23.9.6 BPI_FPI Module Registers 23-262
23.9.7 Interrupt Registers 23-270
23.10 Revision History 23-279

24 Generic Timer Module (GTM) 24-1
24.1 Introduction 24-1
24.1.1 Overview 24-1
24.1.2 Document Structure 24-1
24.2 GTM Architecture 24-3
24.2.1 Overview 24-3
24.2.1.1 GTM Architecture Block Diagram 24-3
24.2.1.2 GTM signal multiplex 24-4
24.2.2 GTM Interfaces 24-7
24.2.2.1 GTM Generic Bus Interface (AEI) 24-7
24.2.2.2 GTM Multi-master and multi-tasking support 24-8
24.2.3 GTM Clock and Time Base Management (CTBM) 24-9
24.2.3.1 GTM Clock and time base management architecture 24-9
24.2.4 GTM Interrupt Concept 24-10
24.2.4.1 Level interrupt mode 24-12

24.2.4.2 Pulse interrupt mode . . . 24-14
24.2.4.3 Pulse-notify interrupt mode . . . 24-16
24.2.4.4 Single-pulse interrupt mode . . . 24-18
24.2.4.5 GTM Interrupt concentration method . . . 24-19
24.2.5 GTM Software Debugger Support . . . 24-20
24.2.5.1 Register behaviour in case of Software Debugger accesses . . . 24-20
24.2.6 GTM Programming conventions . . . 24-20
24.2.7 GTM TOP-Level Configuration Registers Overview . . . 24-21
24.2.8 GTM TOP-Level Configuration Registers Description . . . 24-22
24.2.8.1 Register GTM_REV . . . 24-22
24.2.8.2 Register GTM_RST . . . 24-23
24.2.8.3 Register GTM_CTRL . . . 24-24
24.2.8.4 Register GTM_AEI_ADDR_XPT . . . 24-25
24.2.8.5 Register GTM_IRQ_NOTIFY . . . 24-26
24.2.8.6 Register GTM_IRQ_EN . . . 24-27
24.2.8.7 Register GTM_IRQ_FORCINT . . . 24-28
24.2.8.8 Register GTM_IRQ_MODE . . . 24-30
24.2.8.9 Register GTM_BRIDGE_MODE . . . 24-31
24.2.8.10 Register GTM_BRIDGE_PTR1 . . . 24-33
24.2.8.11 Register GTM_BRIDGE_PTR2 . . . 24-34
24.2.8.12 Register GTM_EIRQ_EN . . . 24-35
24.2.8.13 Register GTM_TIM0_AUX_IN_SRC . . . 24-37
24.2.8.14 Register GTM_HW_CONF . . . 24-38
24.3 Clock Management Unit (CMU) . . . 24-40
24.3.1 Overview . . . 24-40
24.3.1.1 CMU Block Diagram . . . 24-41
24.3.2 Global Clock Divider . . . 24-42
24.3.3 Configurable Clock Generation Subunit (CFGU) . . . 24-42
24.3.4 Wave Form of Generated Clock Signal CMU_CLK[x] . . . 24-43
24.3.5 Fixed Clock Generation (FXU) . . . 24-44
24.3.6 External Generation Unit (EGU) . . . 24-44
24.3.7 CMU Configuration Registers Overview . . . 24-45
24.3.8 CMU Configuration Register Description . . . 24-46
24.3.8.1 Register CMU_CLK_EN . . . 24-46
24.3.8.2 Register CMU_GCLK_NUM . . . 24-47
24.3.8.3 Register CMU_GCLK_DEN . . . 24-49
24.3.8.4 Register CMU_CLK_x_CTRL (x:0…5) . . . 24-50
24.3.8.5 Register CMU_CLK_6_CTRL . . . 24-51
24.3.8.6 Register CMU_CLK_7_CTRL . . . 24-52
24.3.8.7 Register CMU_ECLK_z_NUM (z:0…2) . . . 24-53
24.3.8.8 Register CMU_ECLK_z_DEN (z:0…2) . . . 24-54
24.3.8.9 Register CMU_FXCLK_CTRL . . . 24-55
24.4 Time Base Unit (TBU) . . . 24-56

24.4.1 Overview . . . 24-56
24.4.1.1 TBU Block Diagram . . . 24-57
24.4.2 TBU Time Base Channels . . . 24-57
24.4.2.1 TBU Channel Modes . . . 24-57
24.4.3 TBU Configuration Registers Overview . . . 24-58
24.4.4 TBU Registers description . . . 24-58
24.4.4.1 Register TBU_CHEN . . . 24-59
24.4.4.2 Register TBU_CH0_CTRL . . . 24-60
24.4.4.3 Register TBU_CH0_BASE . . . 24-61
24.4.4.4 Register TBU_CHy_CTRL (y:1, 2) . . . 24-62
24.4.4.5 Register TBU_CHy_BASE (y:1,2) . . . 24-63
24.5 Timer Input Module (TIM) . . . 24-64
24.5.1 Overview . . . 24-64
24.5.1.1 TIM Block Diagram . . . 24-64
24.5.1.2 Input source selection INPUTSRCx . . . 24-65
24.5.1.3 Input Observation . . . 24-66
24.5.1.4 External capture source selection EXTCAPSRCx . . . 24-66
24.5.2 TIM Filter Functionality (FLT) . . . 24-67
24.5.2.1 Overview . . . 24-67
24.5.2.2 TIM Filter Modes . . . 24-69
24.5.2.3 TIM Filter reconfiguration . . . 24-75
24.5.3 Timeout Detection Unit (TDU) . . . 24-75
24.5.3.1 Architecture of the TDU Subunit . . . 24-76
24.5.4 TIM Channel Architecture . . . 24-77
24.5.4.1 Overview . . . 24-77
24.5.4.2 TIM Channel Modes . . . 24-79
24.5.5 TIM Interrupt Signals . . . 24-87
24.5.6 TIM Configuration Registers Overview . . . 24-88
24.5.7 TIM Configuration Registers Description . . . 24-90
24.5.7.1 Register TIM0_CHx_CTRL (x:0…7) . . . 24-91
24.5.7.2 Register TIM0_CHx_FLT_RE(x:0…7) . . . 24-96
24.5.7.3 Register TIM0_CHx_FLT_FE (x:0…7) . . . 24-97
24.5.7.4 Register TIM0_CHx_GPR0 (x:0…7) . . . 24-98
24.5.7.5 Register TIM0_CHx_GPR1 (x:0…7) . . . 24-99
24.5.7.6 Register TIM0_CHx_CNT (x:0…7) . . . 24-100
24.5.7.7 Register TIM0_CHx_CNTS (x:0…7) . . . 24-101
24.5.7.8 Register TIM0_CHx_IRQ_NOTIFY (x:0…7) . . . 24-102
24.5.7.9 Register TIM0_CHx_IRQ_EN (x:0…7) . . . 24-103
24.5.7.10 Register TIM0_CHx_IRQ_FORCINT (x:0…7) . . . 24-104
24.5.7.11 Register TIM0_CHx_IRQ_MODE (x:0…7) . . . 24-105
24.5.7.12 Register TIM0_RST . . . 24-106
24.5.7.13 Register TIM0_IN_SRC . . . 24-107
24.5.7.14 Register TIM0_CHx_EIRQ_EN (x:0…7) . . . 24-110

24.5.7.15 Register TIM0_CHx_TDUV (x:0…7) 24-111
24.5.7.16 Register TIM0_CHx_TDUC (x:0…7) 24-112
24.5.7.17 Register TIM0_CHx_ECNT (x:0…7) 24-113
24.5.7.18 Register TIM0_CHx_ECTRL (x:0…7) 24-114
24.5.7.19 Register TIM0_INP_VAL 24-115
24.6 Timer Output Module (TOM) 24-116
24.6.1 Overview 24-116
24.6.1.1 TOM Block diagram 24-117
24.6.2 TOM Global Channel Control (TGC0, TGC1) 24-118
24.6.2.1 Overview 24-118
24.6.2.2 TGC Subunit 24-118
24.6.3 TOM Channel (TOM_CH[x]) 24-121
24.6.3.1 TOM Channel 0…7 architecture 24-122
24.6.3.2 TOM Channel 8…14 architecture 24-123
24.6.3.3 TOM Channel 15 architecture for PCM generation 24-124
24.6.3.4 Duty cycle, period and selected counter clock frequency update mechanisms 24-126
24.6.3.5 TOM continuous mode 24-128
24.6.3.6 TOM One shot mode 24-129
24.6.3.7 Pulse count modulation 24-131
24.6.4 TOM Interrupt signals 24-132
24.6.5 TOM Configuration Register overview 24-133
24.6.6 TOM Configuration Registers Description 24-135
24.6.6.1 Register TOMi_TGC0_GLB_CTRL 24-135
24.6.6.2 Register TOMi_TGC0_ENDIS_CTRL 24-138
24.6.6.3 Register TOMi_TGC0_ENDIS_STAT 24-140
24.6.6.4 Register TOMi_TGC0_ACT_TB 24-142
24.6.6.5 Register TOMi_TGC0_OUTEN_CTRL 24-143
24.6.6.6 Register TOMi_TGC0_OUTEN_STAT 24-145
24.6.6.7 Register TOMi_TGC0_FUPD_CTRL 24-147
24.6.6.8 Register TOMi_TGC0_INT_TRIG 24-149
24.6.6.9 Register TOMi_CHx_CTRL (x:0…14) 24-150
24.6.6.10 Register TOMi_CH15_CTRL 24-154
24.6.6.11 Register TOMi_CHx_CN0 (x:0…15) 24-157
24.6.6.12 Register TOMi_CHx_CM0 (x:0…15) 24-157
24.6.6.13 Register TOMi_CHx_SR0 (x:0…15) 24-159
24.6.6.14 Register TOMi_CHx_CM1 (x:0…15) 24-159
24.6.6.15 Register TOMi_CHx_SR1 (x:0…15) 24-161
24.6.6.16 Register TOMi_CHx_STAT (x:0…15) 24-161
24.6.6.17 Register TOMi_CHx_IRQ_NOTIFY (x:0…15) 24-162
24.6.6.18 Register TOMi_CHx_IRQ_EN (x:0…15) 24-163
24.6.6.19 Register TOMi_CHx_IRQ_FORCINT (x:0…15) 24-164
24.6.6.20 Register TOMi_CHx_IRQ_MODE (x:0…15) 24-166

24.7 Dead Time Module 24-167
24.7.1 Overview 24-167
24.7.1.1 DTM overview 24-167
24.7.1.2 Connections of TIM and TOM and to DTM 24-168
24.7.2 DTM Channel 24-169
24.7.2.1 DTM channel overview 24-169
24.7.2.2 Standard dead time generation 24-170
24.7.2.3 Wave signals for function of dead time generation 24-171
24.7.3 Phase Shift Control Unit 24-172
24.7.3.1 Phase Shift Unit overview 24-172
24.7.3.2 Example wave of phase shift on channel 1 24-173
24.7.4 Multiple output signal combination 24-174
24.7.4.1 Combination of input signal TIM0_CHi_F_OUT (TIM_CH_IN)/AUX_IN with TOM signal 24-175
24.7.4.2 Combination of multiple TOM output signals 24-175
24.7.4.3 Pulse generation on edge 24-176
24.7.5 Synchronous update of channel control register 2 24-176
24.7.5.1 Synchronous update mechanism of register DTMi_CH_CTRL2 24-176
24.7.6 Configuration Register Overview 24-177
24.7.7 Configuration Register Description 24-178
24.7.7.1 DTMi_CTRL 24-179
24.7.7.2 DTMi_CH_CTRL1 24-181
24.7.7.3 DTMi_CH_CTRL2 24-184
24.7.7.4 DTMi_CH_CTRL2_SR 24-188
24.7.7.5 DTMi_PS_CTRL 24-192
24.7.7.6 DTMi_CHx_DTV 24-194
24.8 Interrupt Concentrator Module (ICM) 24-195
24.8.1 Overview 24-195
24.8.2 Bundling 24-195
24.8.2.1 GTM Infrastructure Interrupt Bundling 24-195
24.8.2.2 TIM Interrupt Bundling 24-195
24.8.2.3 TOM Interrupt Bundling 24-196
24.8.3 ICM Interrupt Signals 24-199
24.8.4 ICM Configuration Registers Overview 24-200
24.8.5 ICM Configuration Registers Description 24-201
24.8.5.1 Register ICM_IRQG_0 24-201
24.8.5.2 Register ICM_IRQG_2 24-202
24.8.5.3 Register ICM_IRQG_6 24-204
24.8.5.4 Register ICM_IRQG_MEI (Module Error Interrupt) 24-207
24.8.5.5 Register ICM_IRQG_CEI1 (Channel Error Interrupt 1) 24-208
24.9 GTM Implementation 24-210
24.9.1 GTM Registers 24-210
24.9.2 Port Connections 24-219

24.9.2.1 Port to GTM Control Registers 24-234
24.9.2.2 GTM to Port Control Registers 24-250
24.9.3 ADC Connections 24-251
24.9.4 SENT Connections 24-253
24.9.5 CAN Connection 24-254
24.9.6 CCU6x Connections 24-256
24.9.7 SCU Connections 24-256
24.9.8 GTM Debug Interface 24-257
24.9.8.1 OCDS Trigger Bus (OTGB) Interface 24-257
24.9.8.2 GTM Debug Registers 24-260
24.10 Revision History 24-265

25 Capture/Compare Unit 6 (CCU6) 25-1
25.1 Introduction 25-1
25.1.1 Feature Set Overview 25-2
25.1.2 Block Diagram 25-3
25.1.3 CCU6 Kernel Registers 25-4
25.2 Operating Timer T12 25-8
25.2.1 T12 Overview 25-9
25.2.2 T12 Counting Scheme 25-11
25.2.2.1 Clock Selection 25-11
25.2.2.2 Edge-Aligned / Center-Aligned Mode 25-12
25.2.2.3 Single-Shot Mode 25-14
25.2.3 T12 Compare Mode 25-15
25.2.3.1 Compare Channels 25-15
25.2.3.2 Channel State Bits 25-16
25.2.3.3 Hysteresis-Like Control Mode 25-21
25.2.4 Compare Mode Output Path 25-22
25.2.4.1 Dead-Time Generation 25-22
25.2.4.2 State Selection 25-24
25.2.4.3 Output Modulation and Level Selection 25-25
25.2.5 T12 Capture Modes 25-27
25.2.6 T12 Shadow Register Transfer 25-31
25.2.7 Timer T12 Operating Mode Selection 25-32
25.2.8 T12 related Registers 25-33
25.2.8.1 T12 Counter Register 25-33
25.2.8.2 Period Register 25-34
25.2.8.3 Capture/Compare Registers 25-35
25.2.8.4 Capture/Compare Shadow Registers 25-36
25.2.8.5 Dead-time Control Register 25-37
25.2.9 Capture/Compare Control Registers 25-39
25.2.9.1 Channel State Bits 25-39
25.2.9.2 T12 Mode Control Register 25-43

25.2.9.3 Timer Control Registers 25-44
25.3 Operating Timer T13 25-53
25.3.1 T13 Overview 25-53
25.3.2 T13 Counting Scheme 25-56
25.3.2.1 Clock Selection 25-56
25.3.2.2 T13 Counting 25-57
25.3.2.3 Single-Shot Mode 25-58
25.3.2.4 Synchronization to T12 25-59
25.3.3 T13 Compare Mode 25-61
25.3.4 Compare Mode Output Path 25-63
25.3.5 T13 Shadow Register Transfer 25-64
25.3.6 T13 related Registers 25-66
25.3.6.1 T13 Counter Register 25-66
25.3.6.2 Period Register 25-67
25.3.6.3 Compare Register 25-68
25.3.6.4 Compare Shadow Register 25-69
25.4 Synchronous Start Feature 25-70
25.5 Trap Handling 25-71
25.6 Multi-Channel Mode 25-73
25.7 Hall Sensor Mode 25-75
25.7.1 Hall Pattern Evaluation 25-76
25.7.2 Hall Pattern Compare Logic 25-78
25.7.3 Hall Mode Flags 25-79
25.7.4 Hall Mode for Brushless DC-Motor Control 25-81
25.8 Modulation Control Registers 25-83
25.8.1 Modulation Control 25-83
25.8.2 Trap Control Register 25-85
25.8.3 Passive State Level Register 25-88
25.8.4 Multi-Channel Mode Registers 25-89
25.9 Interrupt Handling 25-96
25.9.1 Interrupt Structure 25-96
25.9.2 Interrupt Registers 25-98
25.9.2.1 Interrupt Status Register 25-98
25.9.2.2 Interrupt Status Set Register 25-101
25.9.2.3 Status Reset Register 25-103
25.9.2.4 Interrupt Enable Register 25-105
25.9.2.5 Interrupt Node Pointer Register 25-107
25.10 General Module Operation 25-110
25.10.1 Input Selection 25-110
25.10.2 Input Monitoring 25-111
25.10.3 OCDS Suspend 25-112
25.10.4 OCDS Trigger Bus (OTGB) Interface 25-114
25.10.5 General Registers 25-115

25.10.5.1 ID Register . . . 25-115
25.10.5.2 Port Input Select Registers . . . 25-116
25.10.5.3 Module Configuration Register . . . 25-121
25.10.5.4 Input Monitoring Register . . . 25-122
25.10.5.5 Lost Indicator Register . . . 25-125
25.10.5.6 Kernel State Control Sensitivity Register . . . 25-127
25.10.6 BPI Registers . . . 25-128
25.10.6.1 System Registers . . . 25-130
25.11 Implementation . . . 25-138
25.11.1 Address Map . . . 25-138
25.11.1.1 Module Registers . . . 25-139
25.11.2 Module Output Select . . . 25-142
25.11.3 Synchronous Start . . . 25-144
25.11.4 Digital Connections . . . 25-145
25.11.4.1 Connections of CCU60 . . . 25-145
25.11.4.2 Connections of CCU61 . . . 25-150

26 General Purpose Timer Unit (GPT12) . . . 26-1
26.1 Timer Block GPT1 . . . 26-2
26.1.1 GPT1 Core Timer T3 Control . . . 26-4
26.1.2 GPT1 Core Timer T3 Operating Modes . . . 26-6
26.1.3 GPT1 Auxiliary Timers T2/T4 Control . . . 26-13
26.1.4 GPT1 Auxiliary Timers T2/T4 Operating Modes . . . 26-14
26.1.5 GPT1 Clock Signal Control . . . 26-25
26.1.6 GPT1 Registers . . . 26-28
26.1.6.1 GPT1 Timer Registers . . . 26-28
26.1.6.2 GPT1 Timer Control Registers . . . 26-30
26.2 Timer Block GPT2 . . . 26-40
26.2.1 GPT2 Core Timer T6 Control . . . 26-42
26.2.2 GPT2 Core Timer T6 Operating Modes . . . 26-44
26.2.3 GPT2 Auxiliary Timer T5 Control . . . 26-47
26.2.4 GPT2 Auxiliary Timer T5 Operating Modes . . . 26-48
26.2.5 GPT2 Register CAPREL Operating Modes . . . 26-53
26.2.6 GPT2 Clock Signal Control . . . 26-59
26.2.7 GPT2 Registers . . . 26-62
26.2.7.1 GPT2 Timer Registers . . . 26-62
26.2.7.2 GPT2 Timer Control Registers . . . 26-64
26.3 GPT12 Kernel Register Overview . . . 26-71
26.4 General Module Operation . . . 26-72
26.4.1 Input Selection . . . 26-72
26.4.2 OCDS Suspend . . . 26-72
26.4.3 Miscellaneous GPT12 Registers . . . 26-73
26.4.3.1 Port Input Select Register . . . 26-73

26.4.3.2 Identification Register 26-74
26.4.4 BPI Registers 26-76
26.4.4.1 System Registers 26-78
26.5 Implementation of the GPT12 Module 26-85
26.5.1 Address Map 26-85
26.5.2 Module Connections 26-85

27 Versatile Analog-to-Digital Converter (VADC) 27-1
27.1 Introduction and Basic Structure 27-4
27.2 Electrical Models 27-9
27.3 Configuration of General Functions 27-14
27.3.1 Module Identification 27-14
27.3.2 System Registers 27-15
27.3.3 General Clocking Scheme and Control 27-22
27.3.4 Register Access Control 27-27
27.4 Analog Module Activation and Control 27-30
27.4.1 Analog Converter Control 27-30
27.4.2 Calibration 27-30
27.5 Conversion Request Generation 27-32
27.5.1 Queued Request Source Handling 27-34
27.5.2 Channel Scan Request Source Handling 27-64
27.6 Request Source Arbitration 27-80
27.6.1 Arbiter Operation and Configuration 27-82
27.6.2 Conversion Start Mode 27-88
27.7 Analog Input Channel Configuration 27-90
27.7.1 Channel Parameters 27-90
27.7.2 Alias Feature 27-96
27.7.3 Conversion Modes 27-98
27.7.4 Compare with Standard Conversions (Limit Checking) 27-100
27.7.5 Utilizing Fast Compare Mode 27-102
27.7.6 Boundary Flag Control 27-103
27.8 Conversion Timing 27-109
27.9 Conversion Result Handling 27-110
27.9.1 Storage of Conversion Results 27-110
27.9.2 Data Alignment 27-121
27.9.3 Wait-for-Read Mode 27-122
27.9.4 Result FIFO Buffer 27-123
27.9.5 Result Event Generation 27-124
27.9.6 Data Modification 27-125
27.10 Synchronization of Conversions 27-132
27.10.1 Synchronized Conversions for Parallel Sampling 27-132
27.10.2 Equidistant Sampling 27-137
27.11 Safety Features 27-139

27.11.1 Broken Wire Detection 27-139
27.11.2 Signal Path Test Modes 27-140
27.11.3 Configuration of Test Functions 27-142
27.11.4 Automatic Execution of Test Sequences 27-145
27.12 External Multiplexer Control 27-148
27.13 Service Request Generation 27-153
27.14 Implementation into the TC21x/TC22x/TC23x 27-167
27.14.1 Product-Specific Configuration 27-167
27.14.2 Summary of Registers and Locations 27-169
27.14.3 Analog Module Connections in the TC21x/TC22x/TC23x 27-174
27.14.4 Digital Module Connections in the TC21x/TC22x/TC23x 27-176
27.15 Use Case Example for VADC 27-182

28 Input Output Monitor (IOM) 28-1
28.1 Overview 28-1
28.2 Features 28-1
28.3 Interfaces 28-2
28.4 Kernel Description 28-3
28.5 Filter & Prescaler Channel Description 28-4
28.6 EXOR Combiner Description 28-12
28.7 Logic Analyzer Module (LAM) Description 28-13
28.8 Event Combiner Module (ECM) Description 28-15
28.9 IOM Registers 28-17
28.9.1 IOM Identification Register (IOM_ID) 28-20
28.9.2 Filter & Prescaler Cell (FPC) Registers 28-21
28.9.3 GTM Input Related Registers 28-25
28.9.4 Logic Analyzer Module (LAM) Registers 28-27
28.9.5 Event Combiner Module (ECM) Registers 28-32
28.9.6 System Registers 28-37
28.10 SoC Integration 28-44
28.11 Example Monitor/Safety Measures 28-47
28.11.1 Example 1 - Pulse or duty cycle too short 28-48
28.11.2 Example 2 - Pulse or duty cycle too long 28-49
28.11.3 Example 3 - Period too short 28-50
28.11.4 Example 4 - Period too long 28-51
28.11.5 Example 5 - Diagnosis of Command and Feedback - acceptable propagation window and/or signal consistency check 28-52
28.11.6 Example 6 - Diagnosis of Set-up and Hold times 28-53

29 Advanced Driver Assistance Subsystem (ADAS) 29-1
29.1 FFT Engine (FFT) 29-3
29.1.1 Overview 29-3
29.1.2 Feature List 29-4
29.1.3 FFT Engine Performance 29-4

29.1.4 System Performance 29-5
29.1.4.1 Data Movement Optimisation 29-5
29.1.4.2 Double Buffering Optimisation 29-6
29.1.5 Address Spaces and Access Protection 29-6
29.1.5.1 Register Address Space 29-6
29.1.5.2 Data Space 29-6
29.1.5.3 Window Coefficient Space 29-7
29.1.6 Data Value Formats 29-7
29.1.6.1 Time and Frequency Sample Format 29-8
29.1.6.2 Window Coefficient Format 29-8
29.1.7 Window Usage 29-8
29.1.7.1 Window Parameters 29-9
29.1.8 Interrupt and DMA Operation 29-9
29.1.9 FFT Engine Registers 29-11
29.1.9.1 FFT Clock Control Register 29-12
29.1.9.2 FFT Module Identification Register 29-12
29.1.9.3 FFT Control and Status Register 29-13
29.1.9.4 FFT History Registers 29-16
29.1.9.5 FFT Debug Control Registers 29-18
29.1.9.6 FFT Kernel Reset Registers 29-22
29.2 Extended Memory (EMEM) 29-25
29.2.1 Block Diagram 29-25
29.2.1.1 Extended Application Memory (XAM) 29-25
29.2.1.2 Family Overview 29-25
29.2.2 Operational Overview 29-26
29.2.3 Extended Memory 29-27
29.2.4 Access Error Generation 29-27
29.2.5 Tiles and RAM Address Ranges 29-27
29.2.6 EMEM Initialization 29-29
29.2.7 Usage Constraints 29-29
29.2.7.1 Clock Constraints 29-29
29.2.7.2 BBB Interface 29-29
29.2.8 Power Saving 29-29
29.2.9 EMEM Configuration 29-29
29.2.10 EMEM Interfaces 29-30
29.2.10.1 LMU Interface 29-30
29.2.10.2 BBB Slave Interface 29-31
29.2.11 Feed-Through Transactions 29-31
29.2.11.1 EMEM Register Accesses 29-31
29.2.12 Address Map 29-31
29.2.13 EMEM Register Description 29-33

30 Ethernet MAC (ETH) 30-1

Table of Contents

30.1 Overview . . . 30-1
30.1.1 General Module Description . . . 30-2
30.1.2 System Overview . . . 30-3
30.1.2.1 System-Level Block Diagram . . . 30-3
30.1.2.2 Interfaces . . . 30-3
30.1.2.3 Transmit and Receive FIFOs . . . 30-3
30.1.3 Features List . . . 30-5
30.1.3.1 GMAC Core Features . . . 30-5
30.1.3.2 DMA Block Features . . . 30-6
30.1.3.3 Transaction Layer (MTL) Features . . . 30-6
30.1.3.4 Monitoring, Test, and Debugging Support Features . . . 30-8
30.2 Architecture . . . 30-9
30.2.1 Introduction . . . 30-9
30.2.2 IEEE 1588-2002 Overview . . . 30-10
30.2.2.1 Reference Timing Source . . . 30-12
30.2.2.2 Transmit Path Functions . . . 30-12
30.2.2.3 Receive Path Functions . . . 30-13
30.2.2.4 Time Stamp Error Margin . . . 30-13
30.2.2.5 Frequency Range of Reference Timing Clock . . . 30-13
30.2.2.6 Advanced Time Stamp Feature Support . . . 30-14
30.2.3 AHB Application Host Interface . . . 30-24
30.2.4 DMA Controller . . . 30-26
30.2.4.1 Initialization . . . 30-27
30.2.4.2 Transmission . . . 30-30
30.2.4.3 Reception . . . 30-35
30.2.4.4 Interrupts . . . 30-39
30.2.5 MAC Transaction Layer (MTL) . . . 30-40
30.2.5.1 Transmit Path . . . 30-40
30.2.5.2 Receive Path . . . 30-45
30.2.6 GMAC Core . . . 30-48
30.2.6.1 Transmission . . . 30-48
30.2.6.2 MAC Transmit Interface Protocol . . . 30-52
30.2.6.3 Reception . . . 30-52
30.2.6.4 System Time Register Module . . . 30-60
30.2.7 MAC Management Counters . . . 30-63
30.2.7.1 Address Assignments . . . 30-63
30.2.7.2 MMC Register Description . . . 30-72
30.2.8 Power Management Block . . . 30-86
30.2.8.1 PMT Block Description . . . 30-86
30.2.8.2 Remote Wake-Up Frame Detection . . . 30-90
30.2.8.3 Magic Packet Detection . . . 30-90
30.2.8.4 System Considerations During Power-Down . . . 30-91
30.2.9 Station Management Agent . . . 30-92

30.2.9.1 Functions . . . 30-92
30.2.9.2 MII Management Write Operation . . . 30-93
30.2.9.3 MII Management Read Operation . . . 30-94
30.2.10 Reduced Media Independent Interface . . . 30-95
30.2.10.1 Block Diagram . . . 30-95
30.2.10.2 Block Overview . . . 30-96
30.2.10.3 Transmit Bit Ordering . . . 30-97
30.2.10.4 RMII Transmit Timing Diagrams . . . 30-97
30.2.11 Interrupts From the GMAC Core . . . 30-100
30.3 Register . . . 30-102
30.3.1 Register Maps . . . 30-103
30.3.1.1 Register Overview . . . 30-103
30.3.1.2 DMA Register Map . . . 30-103
30.3.1.3 GMAC Register Map . . . 30-105
30.3.1.4 Ethernet MAC Additional Module Control Registers . . . 30-111
30.3.2 Register Description . . . 30-112
30.4 Descriptors . . . 30-419
30.4.1 Normal Descriptor Formats . . . 30-419
30.4.1.1 Receive Descriptor . . . 30-420
30.4.1.2 Transmit Descriptor . . . 30-427
30.4.1.3 Descriptor Format With IEEE 1588 Time Stamping Enabled . . . 30-434
30.4.2 Alternate or Enhanced Descriptors . . . 30-438
30.4.2.1 Transmit Descriptor . . . 30-438
30.4.2.2 Receive Descriptor . . . 30-445
30.5 Ethernet MAC Module Implementation . . . 30-454
30.5.1 Interface Connections of the Ethernet MAC Module . . . 30-454
30.5.1.1 On-Chip Connections . . . 30-456
30.5.1.2 Clocks . . . 30-456
30.5.2 Ethernet MAC Module-Implementation Related Registers . . . 30-458
30.5.2.1 Port Control . . . 30-458
30.5.2.2 Clock Control . . . 30-460
30.5.2.3 Additional Register . . . 30-461
30.6 Revision History . . . 30-470

About this Document

This User's Manual describes the Infineon TC21x/TC22x/TC23x family, a family of 32-bit microcontroller DSPs based on the Infineon TriCore Architecture.

The term "TC21x/TC22x/TC23x" generally refers to the following devices:

- TC21x
- TC22x
- TC23x
- TC23x ADAS

This document is designed to be read primarily by design engineers and software engineers who need a detailed description of the interactions of the TC21x/TC22x/TC23x functional units, registers, instructions, and exceptions.

This TC21x/TC22x/TC23x User's Manual describes the features of the TC21x/TC22x/TC23x with respect to the TriCore Architecture. Where the TC21x/TC22x/TC23x directly implements TriCore architectural functions, this manual simply refers to those functions as features of the TC21x/TC22x/TC23x. In all cases where this manual describes a TC21x/TC22x/TC23x feature without referring to the TriCore Architecture, this means that the TC21x/TC22x/TC23x is a direct implementation of the TriCore Architecture.

Where the TC21x/TC22x/TC23x implements a subset of TriCore architectural features, this manual describes the TC21x/TC22x/TC23x implementation, and then describes how it differs from the TriCore Architecture. Such differences between the TC21x/TC22x/TC23x and the TriCore Architecture are documented in the section covering each such subject.

Note: This document describes the maximum feature set of the TC21x/TC22x/TC23x family. Some features or characteristics may not be available in each device version. For an overview of common and key differentiating features see ***Table 1****.*

Related Documentation

A complete description of the TriCore architecture is found in the document entitled "TriCore Architecture Manual". The architecture of the TC21x/TC22x/TC23x is described separately this way because of the configurable nature of the TriCore specification: Different versions of the architecture may contain a different mix of systems components. The TriCore architecture, however, remains constant across all derivative designs in order to preserve compatibility.

This TC21x/TC22x/TC23x Target Specification together with the "TriCore Architecture Manual" is required to understand the complete TC21x/TC22x/TC23x microcontroller functionality.

Text Conventions

This document uses the following text conventions for named components of the TC21x/TC22x/TC23x:

- Functional units of the TC21x/TC22x/TC23x are given in plain UPPER CASE. For example: “The QSPI supports full-duplex and half-duplex synchronous communication”.
- Pins using negative logic are indicated by an overline. For example: “The external reset pin, $\overline{ESR0}$, has a dual function.”.
- Bit fields and bits in registers are in general referenced as “Module_Register name.Bit field” or “Module_Register name.Bit”. For example: “The Current CPU Priority Number bit field CPU_ICR.CCPN is cleared”. Most of the register names contain a module name prefix, separated by an underscore character “_” from the actual register name (for example, “QSPI0_GLOBALCON”, where “QSPI0” is the module name prefix, and “GLOBALCON” is the kernel register name). In chapters describing the kernels of the peripheral modules, the registers are mainly referenced with their kernel register names. The peripheral module implementation sections mainly refer to the actual register names with module prefixes.
- Variables used to describe sets of processing units or registers appear in mixed upper and lower cases. For example, register name “MOFCRn” refers to multiple “MOFCR” registers with variable n. The bounds of the variables are always given where the register expression is first used (for example, “n = 0-255”), and are repeated as needed in the rest of the text.
- The default radix is decimal. Hexadecimal constants are suffixed with a subscript letter “H”, as in 100_H. Binary constants are suffixed with a subscript letter “B”, as in: 111_B.
- When the extent of register fields, groups register bits, or groups of pins are collectively named in the body of the document, they are represented as “NAME[A:B]”, which defines a range for the named group from B to A. Individual bits, signals, or pins are given as “NAME[C]” where the range of the variable C is given in the text. For example: CFG[2:0] and SRPN[0].
- Units are abbreviated as follows:
  - **MHz** = Megahertz
  - **µs** = Microseconds
  - **kBaud, kbit** = 1000 characters/bits per second
  - **MBaud, Mbit** = 1000000 characters/bits per second
  - **Kbyte, KB** = 1024 bytes of memory
  - **Mbyte**, MB = 1048576 bytes of memory

    In general, the k prefix scales a unit by 1000 whereas the K prefix scales a unit by 1024. Hence, the Kbyte unit scales the expression preceding it by 1024. The kBaud unit scales the expression preceding it by 1000. The M prefix scales by 1,000,000 or 1048576, and µ scales by .000001. For example, 1 Kbyte is 1024 bytes, 1 Mbyte is 1024 × 1024 bytes, 1 kBaud/kbit are 1000 characters/bits

per second, 1 MBaud/Mbit are 1000000 characters/bits per second, and 1 MHz is 1000000 Hz.

- Data format quantities are defined as follows:
  - **Byte** = 8-bit quantity
  - **Half-word** = 16-bit quantity
  - **Word** = 32-bit quantity
  - **Double-word** = 64-bit quantity

Abbreviations and Acronyms

The following acronyms and terms are used in this document:

ADC	Analog-to-Digital Converter
ALU	Arithmetic and Logic Unit
ASCLIN	Asynchronous/Synchronous Serial Controller with LIN
BCU	Bus Control Unit
BROM	Boot ROM & Test ROM
CAN	Controller Area Network
CCU6	Capture Compare Unit 6
CCU	Clock Control Unit
CPS	CPU Slave Interface
CPU	Central Processing Unit
CRC	Cyclic Redundancy Code
CSA	Context Save Area
CSFR	Core Special Function Register
DAP	Device Access Port
DAS	Device Access Server
DCACHE	Data Cache
DFLASH	Data Flash Memory
DMA	Direct Memory Access
DMI	Data Memory Interface
DSPR	Data Scratch Pad RAM
ECC	Error Correction Code
EEC	Emulation Extension Chip
EMI	Electro-Magnetic Interference

E-Ray	Flexray Controller
EVR	Embedded Voltage Regulator
FCS	Flash Command State Machine
FM-PLL	PLL with Frequency Modulation support
FPI	Flexible Peripheral Interconnect (Bus protocol)
FPU	Floating Point Unit
GPIO	General Purpose Input/Output
GPR	General Purpose Register
GPT12	General Purpose Timer 12
GTM	Generic Timer Module
PCACHE	Program Cache
I2C	Inter-Integrated Circuit Controller
IFX	Infineon Technologies AG
I/O	Input / Output
IOM	I/O Monitor Unit
JTAG	Joint Test Action Group = IEEE1149.1
LIN	Local Interconnect Network
LMU	Local Bus Memory Unit
MBIST	Memory Build In Self Test
MCHK	Memory Checker module
MMU	Memory Management Unit
MSB	Most Significant Bit
MTU	Memory Test Unit
MultiCAN+	Enhanced Multiple CAN controller
NC	Not Connected
NMI	Non-Maskable Interrupt
OCDS	On-Chip Debug Support
OVRAM	Overlay Memory
PLL	Phase Locked Loop
PCACHE	Program Cache
PFLASH	Program Flash Memory
PMI	Program Memory Interface

PSPR	Program Scratch Pad RAM
QSPI	Queued SPI Controller
RAM	Random Access Memory
RISC	Reduced Instruction Set Computing
SBCU	System Peripheral Bus Control Unit
SCU	System Control Unit
SENT	Single Edge Nibble Transmission
SFI	Shared Resource Interconnect to FPI Bus Interface (SFI Bridge)
SFR	Special Function Register
SMU	Safety Management Unit
SPB	System Peripheral Bus (based on FPI protocol)
SPD	Single Pin DAP
SPI	Synchronous Serial Controller
SRAM	Static Data Memory
SRI	Shared Resource Interconnect
SRN	Service Request Node
STM	System Timer
SWD	Supply Watchdog
TC1.6E	TriCore CPU 1.6 (High Efficiency variant)
VADC	Versatile Analog-to-Digital Converter
WDT	Watchdog Timer

1 Introduction

This User's Manual describes the Infineon TC21x/TC22x/TC23x family, a family of 32-bit microcontroller DSPs based on the Infineon TriCore Architecture.

The term "TC21x/TC22x/TC23x" generally refers to the following devices:

- TC21x
- TC22x
- TC23x
- TC23x ADAS

1.1 System Architecture of the TC21x/TC22x/TC23x

The TC21x/TC22x/TC23x combines three powerful technologies within one silicon die, achieving new levels of power, speed, and economy for embedded applications:

- Reduced Instruction Set Computing (RISC) processor architecture
- Digital Signal Processing (DSP) operations and addressing modes
- On-chip memories and peripherals

DSP operations and addressing modes provide the computational power necessary to efficiently analyse complex real-world signals. The RISC load/store architecture provides high computational bandwidth with low system cost. On-chip memory and peripherals are designed to support even the most demanding high-bandwidth real-time embedded control-systems tasks.

Additional high-level features of the TC21x/TC22x/TC23x include:

- Efficient memory organization: instruction and data scratch memories, caches
- Serial communication interfaces – flexible synchronous and asynchronous modes
- DMA Controller – DMA operations and interrupt servicing
- Flexible interrupt system – configurable interrupt priorities and targets
- General-purpose timers
- High-performance on-chip buses
- On-chip debugging and emulation facilities
- Flexible interconnections to external components
- Flexible power-management

The TC21x/TC22x/TC23x is a high-performance microcontroller with one TriCore CPU, program and data memories, buses, bus arbitration, interrupt system, DMA controller and a powerful set of on-chip peripherals. The TC21x/TC22x/TC23x is designed to meet the needs of the most demanding embedded control systems applications where the competing issues of price/performance, real-time responsiveness, computational power, data bandwidth, and power consumption are key design elements.

The TC21x/TC22x/TC23x offers several versatile on-chip peripheral units such as serial controllers, timer units, and analog-to-digital converters. Within the TC21x/TC22x/TC23x, all these peripheral units are connected to the TriCore CPU /

system via the System Peripheral Bus (SPB) and the Shared Resource Interconnect (SRI). Several I/O lines on the TC21x/TC22x/TC23x ports are reserved for these peripheral units to communicate with the external world.

1.1.1 TC21x/TC22x/TC23x Block Diagram

Figure 1 shows the block diagram of the TC21x/TC22x/TC23x product device (PD) and the TC23x ADAS device. Please note that not all features that are shown in the block diagram are available in all TC21x/TC22x/TC23x variants.

Emulation devices (ED) contain additional components. A block diagram can be found in separate documentation.

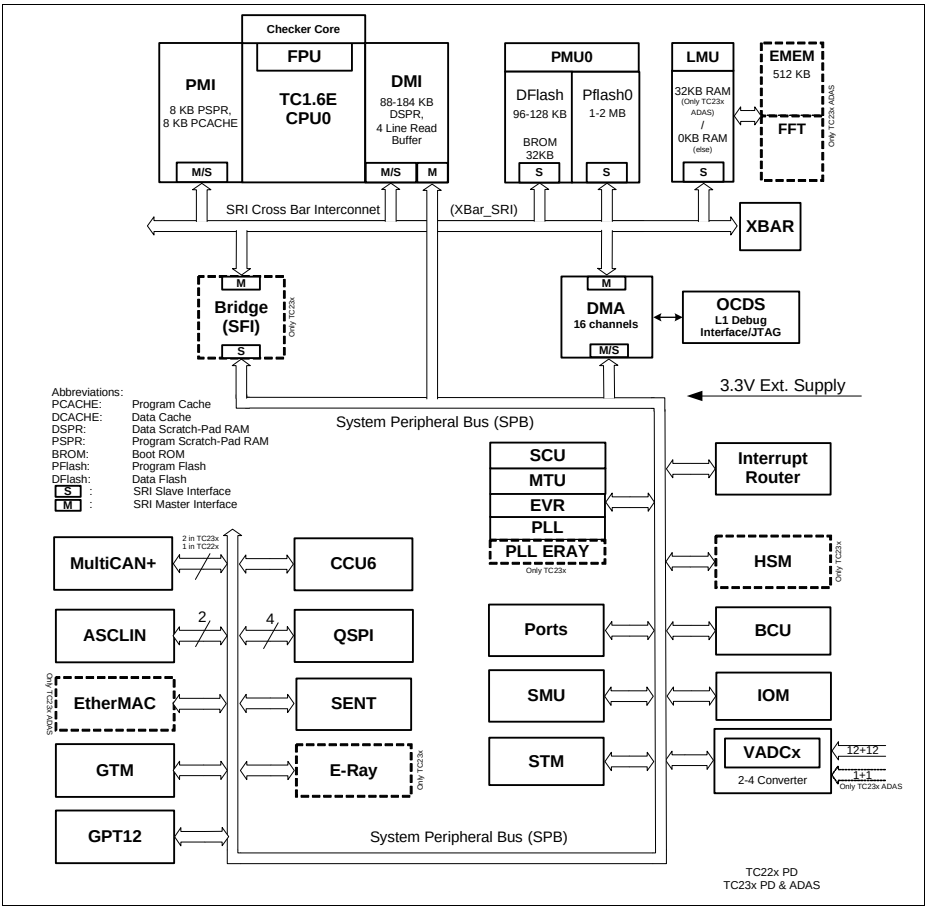


Figure 1 TC21x/TC22x/TC23x Block Diagram

1.1.2 Device Variants

Table 1 contains a comparative overview of the devices of the TC21x/TC22x/TC23x Family including TC21x as derivate of the TC22x. Additional derivates with different feature set might be specified. The feature set of a certain derivate is documented in its Data Sheet or Data Sheet Addendum.

Table 1 Overview of TC21x/TC22x/TC23x Family Devices

Feature		TC21x	TC22x	TC23x	TC23x ADAS
CPU Core	Type	TC1.6E			
	Cores / Checker Cores	1 / 1[1)]		1 / 1	
	Max. Freq.	133 MHz		200 MHz	
	FPU	yes			
	MPU	yes			
Program Flash	Size	512 Kbyte	1 Mbyte	2 Mbyte	
Data Flash	Size	64 Kbyte	96 Kbyte	128 Kbyte	
Cache	Instruction	8 Kbyte			
	Data	4 line read buffer			
SRAM	Size (DSPR/PSPR)	48 / 8 Kbyte	88 / 8 Kbyte	184 / 8 Kbyte	
	Size LMU	0 Kbyte			32 Kbyte
	Size EMEM	0 Kbyte			512 Kbyte
DMA	Channels	16			
ADC (SAR)	Voltage range	0 … 5.5 V (determined by ADC supply)			
	Inputs	24			
	Converters	2			4
	Resolution	12 Bits			
	Channels	12 + 12			12 + 12 + overlay 1 + 1

Table 1 Overview of TC21x/TC22x/TC23x Family Devices (cont'd)

Feature		TC21x	TC22x	TC23x	TC23x ADAS
GTM	TIM	1 (8 input channels)			
	TOM	2 (in sum up to 32 PWM channels)			
	ATOM / MCS	0 / 0			
	CMU / ICM	1 / 1			
	PSM	0			
	TBU	1			
	SPE	0			
	CMP / MON	0 / 0			
	BRC / DPLL	0 / 0			
	DTM	2			
Timer	GPT12	1			
	CCU6	1 module with 2 kernels			
STM	Modules	1			
FlexRay	Modules	–		1	
	Channels	–		2	
CAN	Modules	1		2	
	Nodes	3		3 per module	
	Message Objects	128		128 per module	
	CAN FD	yes		yes (in both modules)	
	Debug over CAN	yes		yes (only in CAN, not in CAN1 module)	
QSPI	Modules	4			
	Speed	up to 20 MBaud on dedicated pins (4 interfaces)			
ASCLIN	Interfaces	2			
SENT	Channels	4			
Ethernet	Channels	0			1
ASIL	Level	up to ASIL-D			
Safety support	SMU	1			
	IOM	1			

Table 1 Overview of TC21x/TC22x/TC23x Family Devices (cont'd)

Feature		TC21x	TC22x	TC23x	TC23x ADAS
FFT		0			1
HSIC (High Speed Input Capture)	Channels	2			
Security	HSM	0		1	
Embedded Voltage Regulator	Type	LDO from 3.3 V to 1.3 V		LDO and switched capacitor DCDC from 3.3 V to 1.3 V	
Low Power Feature	Standby RAM	DSPR			
	Wakeup Timer	yes			
Packages	Type (Pitch)	TQFP-80, TQFP-100, TQFP-144 (all 0.4mm)	TQFP-80, TQFP-100, TQFP-144 (all 0.4mm)	TQFP-100, TQFP-144 (all 0.4mm), LFBGA-292 (0.8mm)	TQFP-144 (0.4mm) LFBGA-292 (0.8mm)
I/O	Voltage	3.3V CMOS (5V input supported on ADC pins)			
$T_{ambient}$	Range (SAK)	−40 … +125°C			
	Range (SAL)	−40 … +150°C			
Clocking	System PLL with FM capability	yes			
	ERAY PLL	no		yes	
	Internal Load Caps C1/C2	None implemented.			

1) Some derivates without checker core.

2 Package Characteristics

This chapter gives a short overview about the package concepts and characteristics of the AURIX microcontroller family. Further package related parameters and characteristics may also be found in the corresponding sections of chapter “Electrical Specification” and/or in the Data Sheet.

The AURIX devices will be based on a family concept regarding compatible port mapping and compatible pinout / ballout between family devices in the same package. Both QFP packages (for devices with lower pin count) as well as BGA packages (for devices with higher pin count) and bare die variants will be offered according to customer requirements.

2.1 Feature List

- Lead free packages
- QFP and BGA package variants (BGA currently not planned for TC22x)
- Moisture Level 3 (JEDEC) or better
- Bare die versions (currently not planned for TC21x/TC22x/TC23x)
- EMC and ESD behavior according to automotive requirements

3 On-Chip System Buses and Bus Bridges

The TC21x/TC22x/TC23x has two independent on-chip buses:

- Shared Resource Interconnect (SRI)
- System Peripheral Bus (SPB)

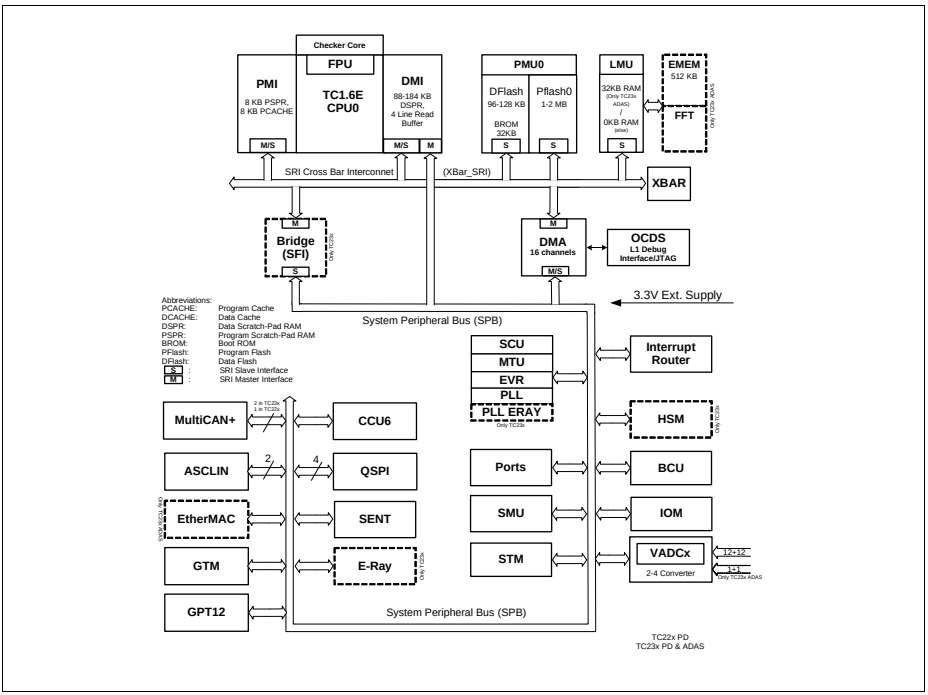


Figure 3-1 On Chip Buses in TC21x/TC22x/TC23x Processor Subsystem

The SRI connects the TC1.6 CPU , the main high bandwidth peripherals and the general purpose DMA module to its local resources for instruction fetches and data accesses.

The System Peripheral Bus connects the TC1.6 CPU and the general purpose DMA module to the medium and low bandwidth peripherals.

Note: The TC1.6 has one SRI slave interface that provides access to the TC1.6 SRAMs, SFRs and CSFRs.

3.1 What is new

Major differences of the TC21x/TC22x/TC23x XBar Bus System architecture compared to previous TC1.6 based products:

- Empty list (This is the initial version of AURIX Products)

3.2 SRI Crossbar (XBar_SRI)

3.2.1 Introduction

The Shared Resource Interconnection (SRI) is the high speed system bus for TriCore1.6.x CPU based devices. The central module of the interconnect is the XBar_SRI which connects all components in one SRI system. The XBar_SRI handles, arbitrates and forwards the communication between all connected SRI-Master and SRI-Slave peripherals.

The XBar_SRI supports parallel transaction between different SRI-Master and SRI-Slave peripherals. It supports also pipe lined requests from the SRI-Master interfaces and pipeline address phases to the connected SRI-Slave interfaces.

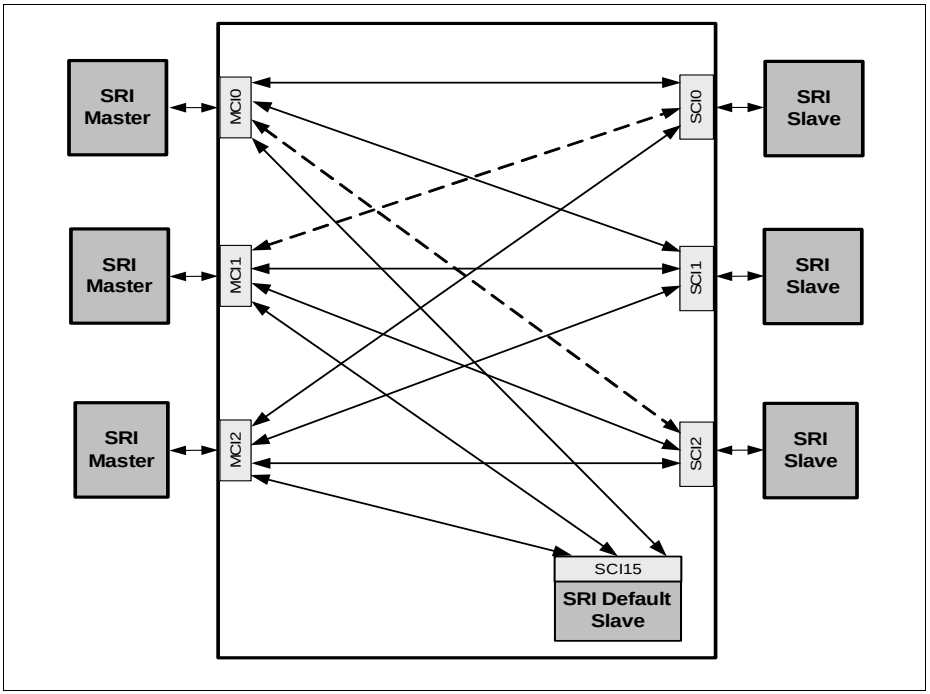


Figure 3-2 XBAR_SRI point to point connection scheme

The XBar_SRI provides SRI Slave Interfaces (SCIx) to connect SRI Slave modules and SRI Master Interfaces (MCIx) to connect SRI Master models to the XBar_SRI. The XBar_SRI includes an Default SRI Slave that provides access to the XBar_SRI control

registers and that takes over all SRI transactions to address outside the connected SRI Slave address ranges. The XBar_SRI includes also one Arbiter module per connected SRI slave module and the infrastructure for the enabled read/write data paths. Each connected SRI slave module as well as the default slave have an related arbiter module within the XBar_SRI.

Please note that only these SRI-Master <-> SRI-Slave connections are implemented that are required for the system functionality (example: connection between PMI Master and PMI Slave is not implemented, see also **Table 3-7**).

For performance optimization the XBar_SRI includes arbitration schemes that allows to configure SRI master priorities for each SRI slave individually (arbiter functionality). For debug support on system level the XBar_SRI includes debug support for SRI-Error and SRI-Transaction ID errors (local SRI slave module support in the related arbiter, global control in the Default Slave) .

Table 3-1 SRI Bus Terms

Term	Description
Agent	An SRI agent is any master or slave device which is connected to the SRI Bus.
Master	An SRI master device is an SRI agent which is able to initiate transactions on the SRI.
Slave	An SRI slave device is an SRI agent which is not able to initiate transactions on the SRI. It is only able to handle operations that are dedicated to it by SRI CrossBar (XBar_SRI).
XBar_SRI	The SRI CrossBar (XBar_SRI) provides the interconnects between connected Master and Slaves. The XBar_SRI includes arbitration mechanisms and debug capabilities. The XBar_SRI has 16 Master Connection Interfaces (MCI0 - MCI15) to connect SRI master devices to it and 15 Slave Connection Interfaces (SCI0 - SCI14) to connect SRI slave devices to it.
MCI	Each Master is connected via one Master Connection Interface (MCI x, x = 0 ...15). The XBar_SRI control registers include control and debug informations related to the Master Connection Interfaces MCI x (x = 0 ... 15).
SCI	Each Slave is connected via one Slave Connection Interface (SCI x, x = 0 ...14). The XBar_SRI control registers include control and debug informations related to the Slave Connection Interfaces SCI x (x = 0 ... 14). The XBar_SRI default slave is connected to SCI 15.

3.2.1.1 XBar_SRI Features

XBar_SRI feature overview:

- Single/Block Data Read Transaction Support (8/16/32/64 Bit)
- Single/Block Data Write Transactions Support (8/16/32/64 Bit)
- Read Modify Write Support
- Supports pipelined requests from SRI master peripherals
- Supports pipelined address phases to SRI slave peripherals
- Single arbiter module for each connected Slave device
- Arbitration priority scheme can be configured for each Slave device individually
- Flexible arbitration schemes (priority, two round robin groups, starvation prevention)
- Breakpoint signal generation based on SRI transactions (OCDS Level 1)
- Configurable MCDS trace interface
- Information integrity support covering SRI address phase signals and the transmitted read/write data
- SRI Address Phase includes Supervisor Mode information (covered by SRI information integrity support)

3.2.2 SRI Transactions

Each SRI transaction consists of:

- one request phase
- one or multiple data phases if not finished with error acknowledge

An SRI master that is requesting for access to an SRI slave is providing all necessary informations about the transaction in parallel with the request. This means that there are no separate on chip bus request and on chip bus address phases. An SRI request/address phase consists of:

- request signal
- lock signal (indicating a read modify write transaction)
- 32-bit address
- 4-bit SRI Op-Code (kind of single data or burst transaction)
- read/write signal (read, write or read modify transaction)
- supervisor mode signal
- Transaction ID (consists of a unique 6-bit Master TAG ID and a 2 bit running number increased for each new transaction of this master)
- 8-bit address phase ECC[1)] (Error Correction Code covering all address phase signals with the exception of request and grant)
- one or multiple data phases if not finished with error acknowledge

1) In the current implementation the Error Correction Code is only used for error detection. Detected errors are reported to the SMU but not corrected.

After the request for a slave access was granted by the related XBar_SRI arbiter module, the transaction can be either finished with error acknowledge or with the read/write data phases as defined during the request/address phase.

Each data phase ends with the transmission of data phase informations including:

- Read or Write Transaction ID (must be equal to the related Transaction ID otherwise the address phase is invalid)
- 8-bit read/write data ECC[1)] (Error Correction Code covering the 64-bit read/write data and the bits [23:3] of the related address.
- 64 bit read or write data. In case of an 8-bit, 16-bit or 32-bit read/write transaction the unused bits are filled with ´don't care´ data. The read/write data ECC[1)] covers the 64 data bits as transferred (including the possible don't care data).

3.2.3 SRI Op-Codes

The SRI Op-Code defines for a SRI transaction the number of data phases, the addressing mode in case of a multi beat transaction and valid bytes in case of a single data transaction.

Table 3-2 Operation Code Encoding

sri_opc[3:0]	Identifier	Description
0000	SDTB	Single Data Transfer Byte (8 bit)
0001	SDTH	Single Data Transfer Half-Word (16 bit)
0010	SDTW	Single Data Transfer Word (32 bit)
0011	SDTD	Single Data Transfer Double-Word (64 bit)
0100	-	Reserved
0101	-	Reserved
0110	-	Reserved
0111	-	Reserved
1000	BTR2	Block Transfer Request (2 transfers) Wrap Around Address Mode is used.
1001	BTR4	Block Transfer Request (4 transfers) Wrap Around Mode is used.
1010	BTRL2[1)]	Block Transfer Request (2 transfers) Linear Address Mode is used.
1011	BTRL4[1)]	Block Transfer Request (4 transfers) Linear Address Mode is used.
1100	-	Reserved

Table 3-2 Operation Code Encoding

sri_opc[3:0]	Identifier	Description
1101	-	Reserved
1110	-	Reserved
1111	-	Reserved

1) The SRI implementation in the TC21x/TC22x/TC23x does not support bursts with linear addressing modes.

3.2.4 SRI Error Conditions

The sri_err_n signal is used by the slave during a transaction to signal the corresponding master that an error has happened which results in an immediate termination of the current transaction. Errors during transaction are tracked by the XBar_SRI and are signalled to via an interrupt and directly to the SMU.

Only the following error conditions are supported and recognized by SRI slaves:

- access level is incorrect (user/supervisor)
- unmapped address access from an SRI master[1)]
- unsupported op-code
- reserved op-code

An SRI error can be generated by the application SW e.g. with an access to a reserved address (see chapter Memory Map):

3.2.5 SRI Transaction ID Error Conditions

Transaction ID is an identifier connected to all phases of a transaction in order to make the transaction unique in the SRI system during the transactions live time.

The transaction ID is used by SRI masters and slaves to identify problems in the SRI system that results in data packets received by a master or slave that do not match the corresponding arbitration/address phase. If the read/write transaction identifier doesn't match with the previous send transaction identifier in the address phase, this is identified as a transaction ID error and tracked by the XBar_SRI.

In situations where at the data source side (master for write transactions, slave for read transactions) a data phase has to be invalidated (e.g. detection of an not correctable SRAM ECC error) and invalid transaction ID is send in order to invalidate the data phase.

An SRI Transaction ID error condition can be generated by injecting an non correctable error into one of the SRAMs (e.g. CPU Instruction Scratch Pad SRAM) and than reading the corrupted data by a CPU.

1) The accesses to unmapped slaves is checked by a default slave.

3.2.6 Operational Overview

This chapter describes the functionality of the XBar_SRI module in order to enable the user to configure the XBar_SRI control registers and to use the XBar_SRI debug resources.

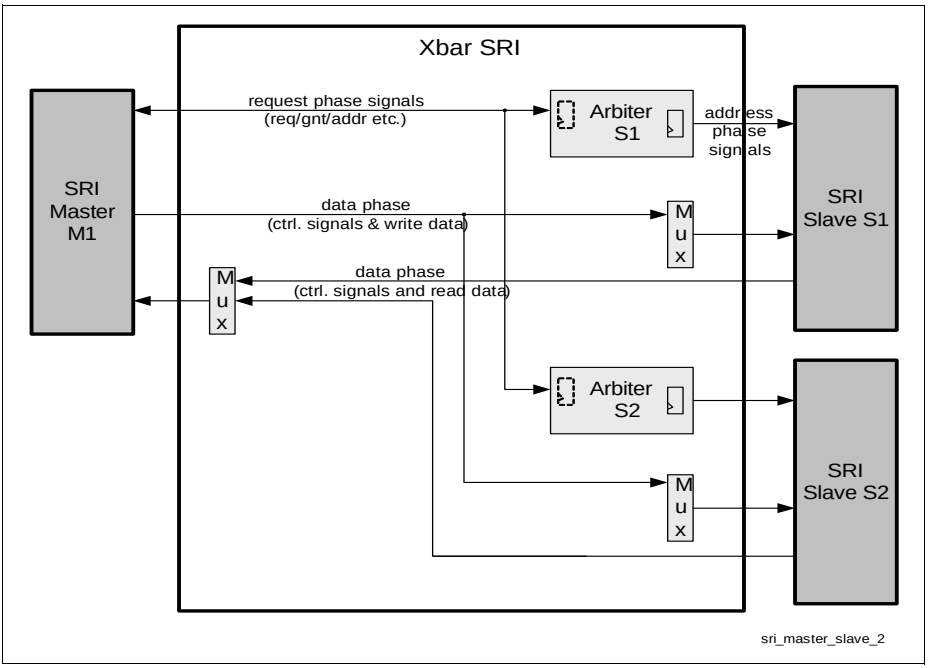


Figure 3-3 XBAR_SRI connections between SRI Master and SRI Slaves

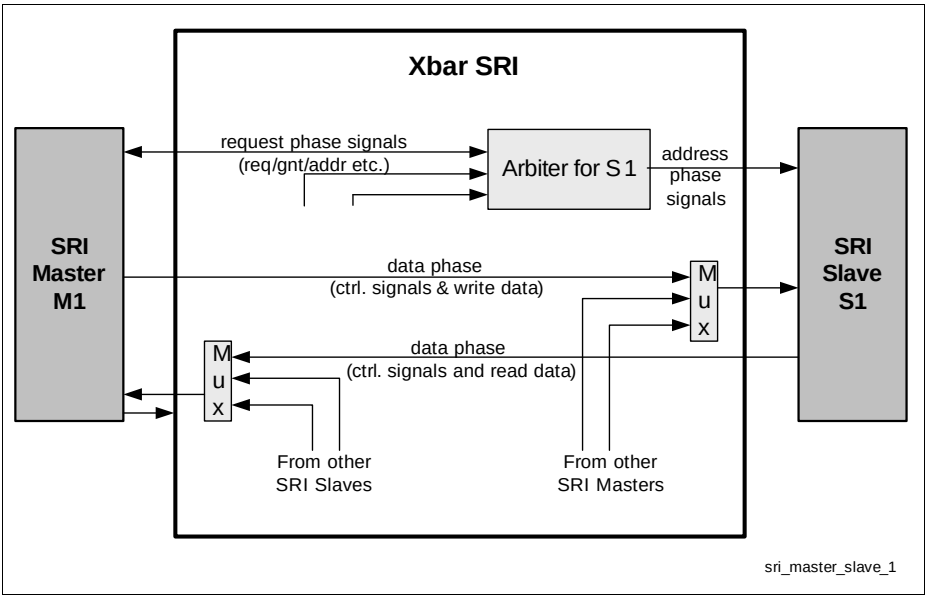


Figure 3-4 XBAR_SRI connections between SRI Slave and SRI Master modules

3.2.6.1 Functional Blocks

The XBar_SRI represents the highest level of the hierarchical SRI-Bus system. Since, it is closest to the TC1.6 core, peripherals critical to CPU performance can be attached to it.

The XBar_SRI module is partitioned into blocks for arbitration and data path control which are necessary for each XBar_SRI master- or slave interface and one block that covers the default slave - and debug functionality.

The XBar_SRI module can handle and process several transaction of different master in parallel if the masters requests different slaves.

XBar_SRI Master Connection Interface (MCI)

Each SRI master module in the system is mapped to one or more XBar_SRI Master Connection Interfaces (MCI). Each MCI is related within the XBAR_SRI module to a default arbitration priority and to register control bits and register control bit fields. So each SRI master in the system is mapped to an XBar_SRI internal default arbitration priority and to master related control register bits and bit fields via its MCI number (see also see also **Table 3-4** and **Table 3-6**).

XBar_SRI Slave Connection Interface (SCI)

Each SRI slave module in the system is mapped to one XBar_SRI Slave Connection Interfaces (SCIx). Each SCI is related within the XBAR_SRI module to one arbiter module with its arbiter module control register and to register control bits and register control bit fields related to this SCIx. So each SRI slave module in the system is mapped to an XBar_SRI internal arbiter module and to slave related control register bits and bit fields via its SCI number (see also see also **Table 3-7**).

XBar_SRI Arbiter

Each XBar_SRI slave connection interface is mapped to exactly one dedicated instantiation of the slave arbiter module in the XBar_SRI. This module includes all required functionality for the following tasks:

- Arbitration: the module includes all required functionality for the arbitration. This includes SRI address decoding, SRI request arbitration, SRI request starvation algorithm, SRI address phase generation and control registers for the priority of the connected master connection interfaces and an FSM to detect the end of the current SRI transaction.
- Debugging: the module tracks the SRI transactions to the dedicated slave connection interface for SRI errors. The transaction information of the first transaction where the SRI protocol error is captured in arbiter internal control registers.
  The module tracks the requests from all masters to detect starvation of masters in the arbitration rounds.
- Error signalling: The first SRI error or starvation error is signalled to the default slave module via two sideband signals.
- XBar_SRI control bus interface: the module has a slave interface to the XBar_SRI control bus. The slave arbiter decodes the address of each new control bus transaction and, if addressed, processes the read/write transaction to the module internal control registers.

Default Slave

The XBar_SRI default slave module is a XBar_SRI internal SRI slave module with its own, dedicated arbiter module inside the XBar_SRI. For accesses to the XBar_SRI control registers only SRI single data transactions of word size are supported. Other SIR op-codes are not supported by the default slave module and acknowledged with an error. The default slave module includes all required functionality for the following tasks:

- As a SRI default slave, it deals with all transactions that are directed to a nonexisting slave in the SRI system as it is described in the SRI protocol. The purpose of the SRI default slave in that situation is to guarantee a defined behavior by terminating these SRI transactions with an error.
- The XBar_SRI default slave module is the SRI interface to all XBar_SRI internal diagnostic- and control registers.

- The XBar_SRI default slave module samples the mci_id_err_n and sci_id_err_n signals from all XBar_SRI master and slave connection interface modules. If the default slave module detects mci_id_err_n and sci_id_err_n pulses it generates an interrupt to the system. The sampling of each mci_id_err_n and sci_id_err_n signal from an arbiter can be disabled via the default slave interrupt control register IDINTEN.
- The XBar_SRI default slave module includes the interrupt node control structure and the corresponding control register.

3.2.7 Functional Overview

3.2.7.1 Arbitration Block

The arbiter has access to all arbitration-/address phase signals from the master connector interfaces he is enabled for, therefore he sees requests from all master connector interfaces in parallel.

In order to check if a master request is addressed to 'its' slave connector interface for the next transaction the arbiter checks if the address transmitted together with the request from the master matches with the allocated address area of this slave via address decoding. Due to the fact that a master can access any slave for a transaction all arbiters of the XBar_SRI are checking the requested address from a requesting master in parallel.

Due to the fact that multiple masters can request for one slave in parallel, each slave has to decode the addresses from all the master connector interfaces it is enabled for in parallel.

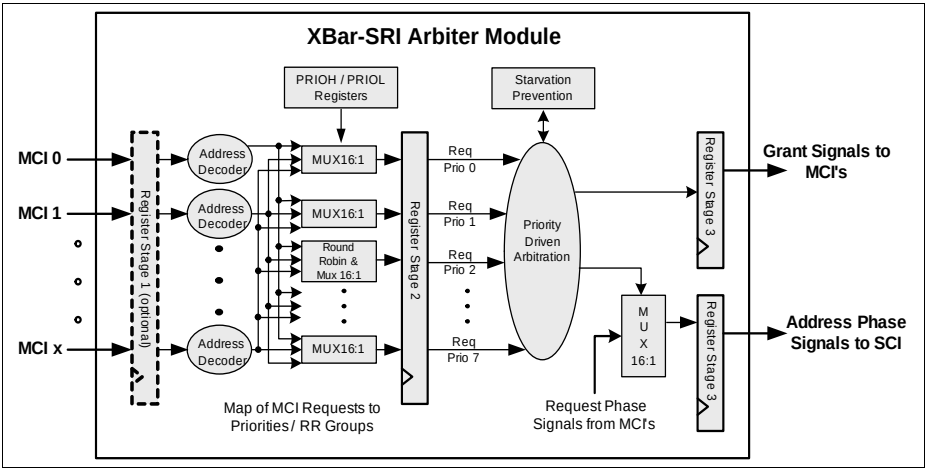


Figure 3-5 XBAR_SRI arbiter scheme

Address Map Checking

The arbiter performs the address comparison for all pending requests from the connected SRI master modules to its slave module in parallel.

Arbitration

A transaction request from a master that matches the address area of a slave connection interface takes part in the next arbitration cycle.

The arbitration is divided into two levels:

- priority driven arbitration
- arbitration within round robing groups (priority 2 and 5)

After reset all enabled master connector interface are mapped to the priorities 2 or 5 (also see also **Table 3-4**). The priorities and the priority algorithm of the master connection interfaces can be programmed for each arbiter individually. The programming of the master priorities can be done by SRI read/write transaction via the default slave module.

The highest priority for an arbiter is 0 and the lowest is 7.

According to the transaction rules described in the SRI protocol specification the arbiter asserts the sri_gnt_x signal to grant the slave to this winning master.

In parallel - or after granting the master, the arbiter sends the address phase for the next transaction to the slave by propagating all necessary informations via the slave connection interface to the slave. The arbiter sends the address phase in parallel with the grant or delayed depending on the address phase pipeline status of the corresponding SRI slave.

For debug purposes, the arbiter samples all necessary transaction informations and provides them to the XBar_SRI default slave module if an SRI error happened and the protocol error feature was enabled.

Arbitration Algorithms

The arbiter related to a slave connection interface (SCI) can be connected to all master connection interfaces (MCI), see also **Chapter 3.2.8.3**. The priority of each MCI can be controlled via the arbiter priority registers (PRIOx). The priority of a master is defined by 3 bit field in the PRIOL / PRIOH register that is related to this master / its MCI number. This can be configured individually for each arbiter so the same master can be handled with different priorities for accesses to different slaves. After reset all enabled MCI are configured with the priority 2 or 5 (Round Robin group).

Please Note:

It must be ensured that two enabled masters don’t have the same priority, with the exception of priority 2 and priority 5 (round robin group priorities).

For changing the priorities during runtime (switching the priorities), all master connection interfaces that should be remapped have to be mapped to the round robin group priorities (2 / 5) before mapping them to their new priorities can be done. This will prevent situations where two masters have the same priority but not a round robin one.

Priority Driven Arbitration

The general arbitration algorithm is priority driven where priority 0 is the highest priority and 7 the lowest one. If multiple masters are requesting for one slave, the master with the highest priority will win the next arbitration round (see also 'starvation prevention').

Round Robin Groups

The arbiter arbitrates in general priority driven, where Priority 2 and priority 5 contain a second 'arbitration' layer. Both priorities can be used as round robin priorities for a group of max. 8 MCI's each. If only one master is mapped to a round robin group priority, the master's request will be treated as normal master request with the priority of the round robin group. If more than one master is mapped to the priority of a round robin group, the requests of the mapped masters will be handled by the round robin algorithm, the winning request of the round robin group arbitration has the priority of the round robin group within the priority driven arbitration.

After the winner of a round robin group is granted, the round robin group starts with a new arbitration round which means the requesting MCI's of the round robin group with the next highest MCI ID number will win the next round robin arbitration round. If there is no requesting MCI with a higher ID number in the round robin group, the algorithm will start with the MCI with the lowest MCI ID number that is mapped to the group, going to the next higher MCI ID number and so on.

Request Latency

If no other request is pending, a request from an MCI has a latency of 1 clock cycle, starting with the detection of the SRI master request on the bus, ending with the SRI grant to the requesting MCI and the SRI address phase to the addressed SCI.

Table 3-3 XBar_SRI Request Latency

Clock Cycle Nr.:	Task(s)
0 (Default Slave)	sri_req_n is sampled at XBar_SRI MCI (Only valid for access to the XBar_SRI control registers)
1	address decoding, mapping of MCI's to priorities / round robin groups and starvation prevention. Round robin arbitrations, priority driven arbitration, mux request phase signals to the address phase registers masking sri_req_n from MCIx after granting the MCIx
2	sri_gnt_n to the requesting MCI, SRI address phase to SCI.

Grant -> New Request Latency (Request Dead Time)

After an MCI was granted, it takes 2 clock cycles before a new request from the same MCI will participate in a new arbitration round even when the SRI master has requested

permanently. This request dead time is a result of the synchronous sri_req_n de-assertion and the XBar_SRI request latency.

Default MCI x Priorities after Reset

After Reset all Master Connection interfaces are mapped to the round robin groups (priority 2 and 5, lower number means higher priority). **Table 3-4** shows the default priority of the MCI with the related coding in the XBAR_PRIOH and XBAR_PRIOL registers. After reset, the priority scheme of the MCIs can be re-configured via the XBAR_PRIOH and XBAR_PRIOL registers, for each SCI (accesses to the Slave connected to an SCI) individually. See also **Chapter 3.2.8.1**,

Note: If multiple CPUs are connected to the Aurix_Bus it is proposed to give the CPU master interfaces (DMI and PMI) the same round robin priority, e.g. 5. This ensures a fair arbitration between CPU access conflicts to the same on chip resource, e.g. Flash.

Table 3-4 Default MCI Priority in the XBar_SRI Arbiters

Priority	Coding	Round Robin Group	Default Mapping:	Comments
0	000	-	-	
1	001	-	-	
2	010	Yes, max 8 master	MCI 0-7	Maximal 8 MCI can be mapped to the priority 2 e.g. can have the priority ’010’.
3	011	-	-	
4	100	-	-	
5	101	Yes, max. 8 master	MCI 8-15	Maximal 8 MCI can be mapped to the priority 8 e.g. can have the priority ’101’.
6	110	-	-	
7	111	-	-	

Starvation Prevention

Starvation can occur when masters with high priority continuously request a dedicated slave, preventing other masters with lower priority from getting access to this slave. To prevent such a situation, a starvation counter has been implemented in each arbiter. This mechanism allows the promotion of weak masters.

Each time the starvation counter in a slave module has and underflow, all pending requests to this arbiter will be entered in the arbiters request list. This even applies to all masters mapped to a round robin group.

Next time when the starvation counter has an underflow, the masters of this request list will be entered in the arbiters request promotion list if:

- The request was not granted during the last starvation period

The masters in that request promotion list will be the next to be granted independent of their priorities. The masters in the request promotion list will be granted one after the other, starting with the master which has the lowest MCI number in this list if there are more than one. The master promotion list has the highest priority within the arbiters main arbitration algorithm, therefore all masters in the promotion list will be granted before the arbiter switches back to its 'normal' arbitration. A master is removed from both lists when it is granted.

If several masters are mapped to a round robin priority, all masters of that round robin arbitration round will be entered in the request list/request promotion list when not granted.

The value controlling the counter period is programmable. After reset the starvation counter has a value of zero. On a starvation counter underflow it is reloaded with the content of the arbiter control register ARBCON.SPC. An underflow occurs in the clock cycle when the counter tries to count down from zero.

The number of arbitration cycles a master must wait for the slave varies. But it's guaranteed to be no more than 2 x ARBCON.SPC until the master is promoted to the request promotion list.

Each time there is an underflow it is checked if there are any masters in the request promotion list that were not granted since the last underflow. This can happen if there are to many/to long transactions to be promoted compared to ARBCON.SPC value. In this case an error is generated via the xcb_sc_err_n signal to the default slave if the feature was enabled (bit ARBCONx.SCERREN set).

If currently a RMW transaction is processed the starvation counter is stopped before the next overflow. The starvation counter is released again after the address phase of the write part is generated or an error for the RMW transaction was received.

If currently a Read-Modify-Write (RMW) transaction is processed by the slave the starvation protection counter is stopped until this RMW is finished.

3.2.7.2 Default Slave

The default slave serves three features in the XBar_SRI implementation:

Control and Configuration Registers Interface

The XBar_SRI default slave module handles all read and write transactions to the debug- and control registers of the XBar_SRI. For this purpose, the default slave module has its own address space which must be mapped into the system address space by an address decoder, provided by the customer.

For a detailed description of the registers see **Chapter 3.2.9**.

Non Existing Addresses

The XBar_SRI internal default slave module has its own arbiter module.

If an SRI master sends a request with a non existing address[1)] to the XBar_SRI, the transaction will be directed to the default slave. The default slave finishes the transaction with error following the SRI protocol rules, which activates the error tracking mechanism of the arbiter. As a result of the error, the default slave module signals this incorrect behavior to the system by generating an interrupt.

A write from an SRI master to a non existing address on the System Peripheral Bus (SPB) will be handled by the Bus Control Unit on the SPB only. A read from an SRI master to a non existing address on the SPB will be handled by the by the Bus Control Unit on the SPB and also captured by the XBar_SRI arbiter as it will see the transaction as read transaction finished with Error Acknowledge.

Error Handling

If an arbiter detects an SRI protocol error during a transaction with a corresponding slave the involved arbiter samples all relevant information of the transaction in the arbiter internal diagnostic registers (ERRADDRx and ERRx) and signals this event via xcb_sri_err_n or xcb_sri_err_d_n sideband signal to the default slave module.

Note: The two error registers ERRx and ERRADDRx in each arbiter are updated with the content of the currently processed transaction in the data phase. The registers are only updated if they are not locked due to a pending protocol error and sri_ready_n was asserted in parallel with sri_err_n deasserted.

The error capture registers in the arbiter can't be changed until the interrupt was acknowledged to the slave arbiter module (set the ARBCON.INTACK) via the default slave.

The stored informations can be read from the default slave module with SRI single data read transactions, the acknowledge can be sent to the default slave via a write to a specific address.

1) non existing address = all Reserved address ranges described in the Memory Map chapter

This can result in a loss of interrupt data information of the same source beyond the first until the write was processed but as all SRI errors must be analyzed and fixed before the product will work, these kind of errors can be analyzed one after the other.

In case that a master or a slave signals an ID error to the XBar_SRI the default slave marks the source of the ID error by setting the according bit in register IDINTSAT if this feature was enabled for that specific master and/or slave interface.

As a result, the default slave generates an interrupt to the system.

Note: While a status bit for a error source is set in either the INTSAT or IDINTSAT register a new error from this particular source doesn't generate a new interrupt.

1. *Each participant in the SRI-Bus system has three interrupt sources; protocol errors, ID errors and time-out due to starvation. All error from the SRI-Bus components are combined in the single SRN of the XBar_SRI.*

Each arbiter has two 32 bit registers containing the error information. This registers are accessed as all other registers, via the default slave module.

The default slave module has one service request node (SRN) to start interrupts for detected SRI errors. Protocol errors, starvation errors as well as ID errors are handled together by this node.

3.2.7.3 Register Access Protection

The XBar_SRI registers are protected by a master TAG ID based access protection mechanism. Each on chip resource with direct or indirect bus master capability has a unique master TAG ID that can be use to identify the master of an on chip bus transaction.

Access Enable:

TAG ID based protection means that on chip bus write access to the XBar_SRI control registers can be disabled for each master TAG ID individually (with the exception of the Access Enable registers itself, which is Safety Endinit protected). For a disabled master TAG ID, write access will be disconnected with error acknowledge, read access will be processed.

Access Enable Registers (XBAR_ACCEN1/0):

-> defines which master TAG ID is allowed to write to XBar_SRI control registers

-> are Safe Endinit write protected

The Access Enable registers (XBAR_ACCEN1/0) are Safe Endinit protected. The Safe Endinit system status is defined by a Watchdog Unit in the System Control Unit (SCU)

After reset, all XBAR_ACCEN1/0 access enable bits and access control bits are enabled, access protection mechanism has to be configured and checked to bring the system in a safe state.

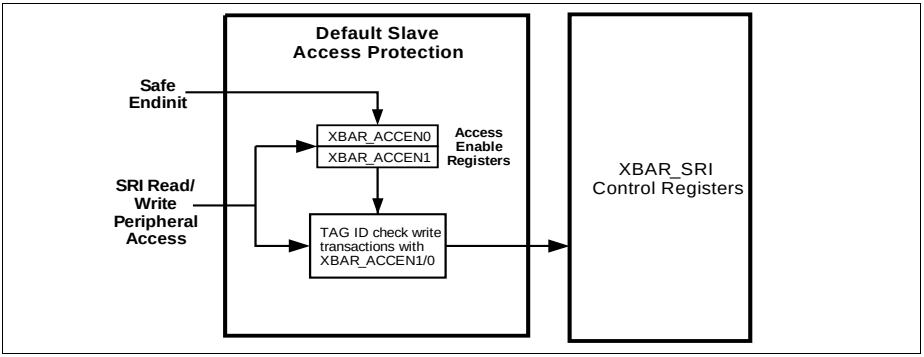


Figure 3-6 XBar_SRI: Control Register Access Protection scheme

3.2.7.4 SRI ECC Error Handling

The SRI protocol provides information integrity support covering:

- The address phase of an SRI transaction
- The transmitted read and write data

The integrity support mechanism is using Error Correction Code (ECC).

Note: In the current implementation the Error Correction Code (ECC) is only used for error detection. Detected errors are reported to the SMU but not corrected.

Detected SRI address phase ECC errors:

- If an SRI slave detects an SRI address phase ECC error it finishes the transaction with SRI error acknowledge and does not further process it (see also **Chapter 3.2.7.5**). Error is signalled to the SMU.

Detected SRI read data ECC errors:

- PMI and DMI: a bus error trap will be generated but only if and when a received 64 bit read data is really used by the CPU. This as PMI / DMI can read complete cache lines (4x64 bit) where the critical 64 bit is used, the other 3 x 64 bit are speculative reads and might not be used by the CPU later.
- DMA: the DMA ignores read data ECC errors but signals the error to the SMU.
- SFI: the SFI SRI master ignores read data ECC errors but signals the error to the SMU

Detected SRI write data ECC errors:

- An detected write data ECC error is signalled to the SMU

- SRI write data ECC errors to SRAMs will not be executed by the related slave modules.

It can happen that a data during a transaction shows an ECC error inside the data source peripheral (master during write, slave during read). Example: A burst read transaction from a memory slave peripheral where the second, third or forth data taken from the SRAM shows inside the slave/master an ECC error. In this situation a slave can finish a read transaction with error acknowledge (if the error is detected before the first data phase) or a slave / master can invalidate the related data phase with an invalid read / write transaction ID error (see also **Chapter 3.2.7.5**).

ECC Code

The ECC codes used for the IR Error Detection mechanism is a Hsiao 22_5 code with DED (double error detection) capability:

- SRI Address Phase informations are protected by the 64_8 code
- SRI Read / Write data are protected by the 96_8 code

```
     CONSTANT     code_matrix     :     matrix_vec96_8     :=(
"0110011011010111111001011011110100010111000001000010001001011001
1000100010101100000010111010101111",
"0001101101101010111010101101111001110111000010000100010010101010
0001000100010101000101011101011",
"1011010110001101010100110110111100111111000100001000100100110100
0000001001011011101000100000111101",
"0001100111110001101111000111101111001111001000010001000110000000
1010010101100010110001000110011110",
"1101111000111110001111111000011110101001010000100001111100000001
0100011010000000111110001111110000",
"1110000000111111110111111111000111001010100000111110000000000010
1111100000000011011111110000000000",
"1110111111000000000111111111111011110000111111100000000000000011
0111111111111100000000000000000000",
"1111111111111111111000000000000011111100111111111111111111111100
0000000000000000000000000000000000");

     CONSTANT     code_matrix     :     matrix_vec64_8     :=     (
"0001011100000100001000100101100100010001010110000001011101010111
1",
"0111011100001000010001001010101000100010001010100010101110101011
1",
"0011111100010000100010010011010000000100101101110100010000111101
1",
```

```
"1100111100100001000100011100000101001010110001011000100011001111
0",
"1010100101000010000111100000000101000110100000001111100011111000
0",
"1100101010000011111000000000001011111000000000011011111110000000
0",
"1111000011111100000000000000001101111111111111000000000000000000
0",
"1111110011111111111111111111110000000000000000000000000000000000
0");
```

3.2.7.5 Error Tracking Capability

The XBar_SRI tracks all SRI transactions for SRI protocol errors. Additionally it tracks informations about starvation errors and SRI transaction ID errors. This is done by all arbiters and the default slave in parallel as the XBar_SRI supports the processing of multiple transactions from master and slaves in parallel which can result in parallel events at different slave connector interfaces.

For this purpose, each arbiter has two error/debug registers where it samples the transaction informations of the transaction where the first protocol error happened.

Note: Protocol errors and debug trigger events can lock the error/debug registers. Only the first event of both sources can lock the registers. All other events are not captured with this two error/debug registers unless the lock was released in the meantime.

Further protocol errors will be ignored by an arbiter that has detected an protocol error until the tracking mechanism is re-activated via the arbiter internal control register. A detected protocol error or starvation error is signalled by the arbiter to the default slave module via two separate sideband signals for SRI- and starvation error. The default slave samples the sideband signals pulses in an error status register INTSAT. As each slave has its own sideband signals, the default slave has the information which arbiter has detected an SRI protocol or starvation error. Each error signal can be masked individually by control registers in the arbiter modules. All debug registers are accessible via the SRI-Bus interface of the default slave.

For transaction ID errors of write transactions the SCI propagates the sci_id_err_n signal via the sri_wr_tr_id_err signal from the slave to the default slave module. The default slave sets the assigned bit in register IDINTSAT.

For transaction ID error of read transactions the MCIs propagate the sri_id_err_n signal via the sri_rd_tr_id_err signal from the master to the default slave module. The default slave sets the assigned bit in register IDINTSAT.

Once an error was signalled from an arbiter or an MCI/SCI to the default slave, the default slave module sends an interrupt request to the system. The system can read out

the error status registers in the default slave module to find out which arbiter(s) or master/slave have detected an error, then the system can start with more detailed diagnostics by reading out the error/debug registers in the arbiter for a protocol error. The error informations captured for an SRI protocol error allows the identification of the master via the sampled master tag ID and the final destination via the sampled target address. Additionally the arbiter samples the op-code and the sri_rd_n, sri_wr_n and sri_svm control signals of the transaction.

In addition to the SRI transaction information, the XBar_SRI captures sideband signal informations (**XBAR_ERR0**.MCI_SBS and **XBAR_ERRD**.MCI_SBS). In the TC21x/TC22x/TC23x these sideband signals are used by the DMA SRI master interface to provide informations about the requestor of a transaction, in parallel to the SRI request phase. **Table 3-5** shows the encoding of the MCI_SBS bit field for the encoding of the TC21x/TC22x/TC23x.

Table 3-5 Encoding of ERRx.MCI_SBS in the TC21x/TC22x/TC23x

MCI_SBS[7:0]	Bit field encoding
MCI_SBS[7:0]	This bit field is only valid if the master TAG ID of the address phases is related to one of the DMA hardware resource groups. MCI_SBS[7:0] is showing the number of the DMA channel that initiated the transaction.

After reading all relevant error informations, the error/debug tracking mechanism can be reactivated.

3.2.7.6 Debug Trigger Event Generation (OCDS Level 1)

This functionality can be used to generate breakpoints on selectable and configurable events in the SRI-Bus traffic.

The debug trigger event generation is controlled via the three debug registers DBCON, DBADD and DBMADD and the three status registers DBSAT, ERR and ERRADDR.

All DBXXX registers reset only with the debug reset (Class 1 reset).

The register DBSAT collects all debug trigger events from all arbiters in the XBar_SRI module. When an arbiter generates a debug trigger event the according bit in register DBSAT is set.

The DBCON register defines the debug trigger event conditions for each arbiter individually. Several individual break conditions can be combined by enabling them in parallel. Possible break conditions are:

- Write transactions
  - If bit DBCON.WREN is set only transactions with an asserted sri_wr_n signal can generate a debug event.

Note: Only if DBCON.WREN and DBCON.RDEN are set together a RMW transaction generate a debug trigger event.

- Read transactions
  – If bit DBCON.RDEN is set only transactions with an asserted sri_rd_n signal can generate a debug event.

Note: Only if DBCON.WREN and DBCON.RDEN are set together a RMW transaction generate a debug trigger event.

- Supervisor mode transactions
  – If bit DBCON.SVMEN is set only transactions with an asserted sri_svm signal can generate a debug event.
- Transactions from a dedicated master
  – If bit DBCON.MASEN is set only transactions initiated by master DBCON.MASTER can generate a debug event.
- Transactions accessing a defined address area
  – If bit DBCON.ADDEN is set only transactions accessing an address in the selected address area can generate a debug event. The selected address area is defined by the registers DBADD and DBMADD. Register DBADD defines one global 32-bit address that is compared with sri_addr[31:0] for all bits where DBMADDR is set to ’1’.

All enabled debug trigger conditions are combined by a logical AND.

Example: If DBCON.WREN and DBCON.SVMEN are set and all other enables are cleared a debug trigger event is only generated for a write transaction operating in the supervisor mode.

Additionally a debug trigger event is generated if DBCON.ERREN is set and an error occurs. Please note that an error occurs only when the generation for this source is enabled in the linked registers ARBCON or IDINTEN.

The result of the logical AND of the first five debug trigger event options is combined with the result of the error debug trigger event by a logical OR.

A debug event is signaled to the default slave. The default slave combines all XBar_SRI arbiter debug event signals with its own and generates the debug event signal that is sent to the OTGM module.

The debug event signal to the OTGM will be asserted for as long at least one condition inside an XBar module or the XBar default slave module is met.

Debug Trigger events inside the XBar_SRI are sampled in the register DBSAT. Additionally an interrupt can be generated for debug trigger events by the XBar_SRI.

When debug condition is reached in one of the XBar_SRI arbiter modules, informations of the transaction that matched to the debug condition is captured in the two error/debug capture registers ERRx and ERRADDRx. If the two registers are already locked due to an earlier action the capturing is not performed.

Writing to DBCON.REARM will rearm the feature, this also sets DBCON.ARM.

3.2.7.7 Interrupt and Debug Events of the XBar_SRI Module

There are some interactions between interrupt them self and debug events. In general due to the nature of the crossbar concept several interrupts from the same or a different source (arbiter, MCI or SCI) can occur. One interrupt could occur several times before a service request routine is initiate. Additionally can all interrupts occur in parallel to one or more debug trigger events.

All following examples assume a time interval without any acknowledge either from a service routine or a debug routine and all consecutive interrupts/events come from the same source.

Two Consecutive Protocol Errors

The first protocol error is captured as normal together with a generation of an interrupt to the system. The second protocol error is not captured as the two registers ERR and ERRADD are already locked and no interrupt is generated.

Two Consecutive Starvation Errors

The first starvation error generate an interrupt to the system. The second starvation error will not generate an interrupt to the system.

Two Consecutive Transaction ID Errors

The first transaction ID error generate an interrupt to the system. The second transaction ID error will not generate an interrupt to the system.

Two Consecutive Debug Trigger Events

The first debug trigger event is captured as normal together with a generation of debug trigger event signal to the system. The second debug trigger event is not captured as the two registers ERR and ERRADD are already locked and a debug trigger event signal is generated.

Protocol Error followed by a Debug Trigger Event

The first protocol error is captured as normal together with a generation of an interrupt to the system. The later debug trigger event is not captured as the two registers ERR and ERRADD are already locked and a debug trigger event signal is generated to the system.

Debug Trigger Event followed by a Protocol Error

The first debug trigger event is captured as normal together with a generation of debug trigger event signal to the system. The later protocol error is not captured as the two registers ERR and ERRADD are already locked and no interrupt is generated.

Starvation/Transaction ID Error followed by a Debug Trigger Event

The first starvation/transaction ID error generate an interrupt to the system. The later debug trigger event is captured to the two registers ERR and ERRADD and a debug trigger event signal is generated to the system.

Debug Trigger Event followed by a Starvation/Transaction ID Error

The first debug trigger event is captured as normal together with a generation of debug trigger event signal to the system. The later starvation/transaction ID error generates an interrupt to the system.

Releasing the Lock from registers ERR and ERRADD

If the ERR and ERRADD registers are locked only from one even only (protocol error or debug trigger event) the lock can be releasing by:

- Writing a one to ARBCONx.INTACK when the registers are locked by a protocol error
- Writing a one to DBCONx.REARM when the registers are locked by a debug trigger event

If both, a protocol error and a debug trigger event occurred since the lock was released the last time both locks have to be released

- Writing a one to DBCONx.REARM AND Writing and to ARBCONx.INTACK when the registers are locked by a debug trigger event AND a protocol error

3.2.8 Implementation of the Cross Bar (XBar_SRI) in the TC21x/TC22x/TC23x

This chapter describes the SRI Interconnect implementation in the TC21x/TC22x/TC23x. The knowledge of the specific implementation (e.g. the connection of the SRI master / slave devices to the Interconnect) is necessary in order to:

- map error informations to the connected slave devices
- define the arbitration scheme for accesses to the connected slave devices
- map XBar_SRI (arbiter) control registers to connected slave devices

This chapter includes three tables that are describing: the relationship (mapping) of

- the relationship (mapping) of TC21x/TC22x/TC23x SRI master devices to the XBar_SRI Master Connection Interfaces MCI 0 - MCI 15 (**Table 3-6**)
- the relationship (mapping) of TC21x/TC22x/TC23x SRI slave devices to the XBar_SRI Slave Connection Interfaces SCI 0 - SCI 15 (**Table 3-7**)

- the point to point connections between TC21x/TC22x/TC23x SRI master and slave devices (**Table 3-7**)

3.2.8.1 Mapping of SRI Master Modules to XBar_SRI Master Interfaces

Table 3-6 shows the mapping of master devices to the XBar_SRI Master Interfaces (MCI 0 - MCI 15). Most of the XBar_SRI control registers are related to the XBar_SRI Slave Interfaces (SCI 0 - SCI 15) or the XBar_SRI Master Interfaces. Therefore it is important to know which TC21x/TC22x/TC23x SRI master device relates to which XBar_SRI Slave Interface.

Example 1:

The XBar_SRI includes error registers where each MCI is represented with 1 bit, showing if during the transfers requested by the master devices connected to the MCI´s an error situation occurred (e.g. **XBAR_IDINTSAT**).

Example 2:

The XBar_SRI includes one arbiter module per connected SRI slave device. Each arbiter module includes a four bit field where the priority requests from connected MCI can be defined. If the access priority of the DMI SRI master to one SRI slave device has to be changed, this can be done via the bit field related to MCI 4 in the arbiter control register related to this SRI slave (control registers: XBAR_PRIOHx, XBAR_PRIOLx, XBAR_PRIODx).

Table 3-6 Mapping of TC21x/TC22x/TC23x SRI master devices to MCI

XBar_SRI Master Connection Interface	MCI Number	Priority after Reset	Connected SRI master device
MCI 0	0	2	DMA
MCI 1	1	-	-
MCI 2	2	-	-
MCI 3	3	-	-
MCI 4	4	-	-
MCI 5	5	TC23x:2 TC22x: -	TC23x: SFI (HSM access to SRI) TC22x: -
MCI 6- MCI11	6-11	-	-
MCI 12	12	5	CPU0.DMI
MCI 13	13	5	CPU0.PMI
MCI 14- MCI 15	14-15	-	-

Note: If multiple CPUs are connected to the Aurix_Bus it is proposed to give the CPU master interfaces (DMI and PMI) the same round robin priority, e.g. 5. This ensures a fair arbitration between CPU access conflicts to the same on chip resource, e.g. Flash.

3.2.8.2 Mapping of SRI Slave modules to XBar_SRI Slave Interfaces

Table 3-7 shows the mapping of slave modules to the XBar_SRI Slave Interfaces (SCI 0 - SCI 15). Most of the XBar_SRI control registers are related to the XBar_SRI Slave Interfaces (SCI 0 - SCI 15) or the XBar_SRI Master Interfaces. Therefore it is important to know which TC21x/TC22x/TC23x SRI slave device relates to which XBar_SRI Slave Interface or arbiter module.

Example 1:

The XBar_SRI includes error registers where each SCI is represented with 1 bit, showing if during the transfers requested by the master devices connected to the MCI´s an error situation occurred (e.g. **XBAR_DBSAT**, **XBAR_IDINTSAT**, **XBAR_IDINTEN**).

Example 2:

The XBar_SRI includes one arbiter module per connected SRI slave device. Each arbiter module includes error capturing resources and breakpoint capabilities. These can be used e.g. to analyze accesses to the connected slave device that where answered with error acknowledge by the slave device (e.g. **XBAR_ERR0**, **XBAR_ERRADDRD**).

Table 3-7 Mapping of TC21x/TC22x/TC23x SRI slave devices to SCI

XBar_SRI Slave Connection Interface (SCI)	Connected SRI master device
SCI 0	CPU0 (PSPR, DSPR, SFR, CSFR, PCache RAM, PTAG)
SCI 1-2	-
SCI 3	-
SCI 4	LMU (EMEM)
SCI 5	-
SCI 6	PMU0: DFlash, BootROM, CTRL Reg
SCI 7	PMU0: PFlash0
SCI 8- SCI 14	-
SCI 15	XBar_SRI Default Slave

3.2.8.3 TC21x/TC22x/TC23x SRI Master / Slave Interconnection Matrix

Table 3-7 shows the SRI master to SRI Slave interconnects that are implemented in the TC21x/TC22x/TC23x (´ ´ in a table cell means connection implemented. ´x´ in a table cell means connection not implemented)

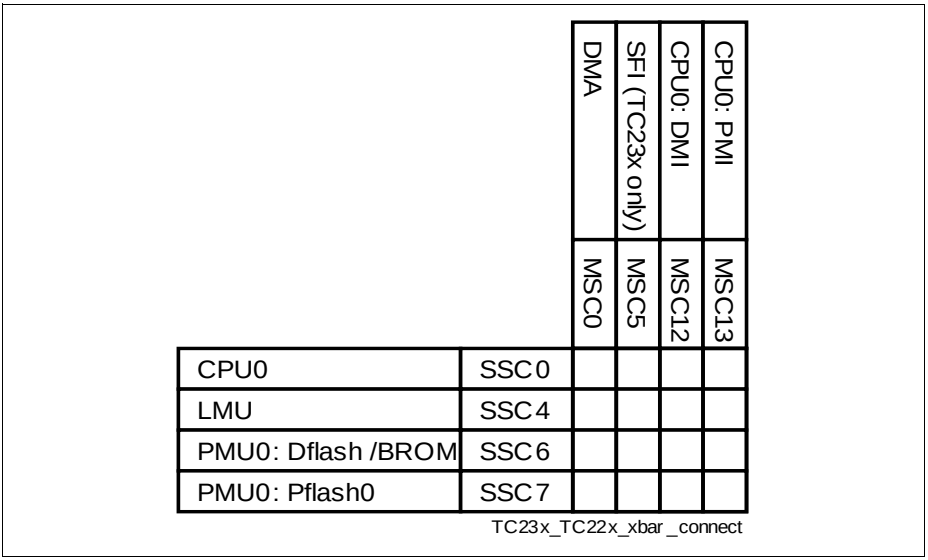


Figure 3-7 TC21x/TC22x/TC23x: SRI Master / Slave Interconnection Matrix

3.2.8.4 Connection Master-Slave in XBar_SRI

As the SRI-Bus protocol is a point to point based bus implementation multiplexer inside the data paths in front of the MCIs and arbiter/SCIs are required (see **Figure 3-3** and **Figure 3-4**).

The write data path multiplexer in front of the SCIs are controlled by the related arbiter modules. The read data path multiplexers in front of the MCIs are controlled by all arbiters.

Master - and Slave side MUX

During an SRI transaction, the corresponding arbiter has to establish the data path connection from his slave connection interface to the corresponding master connection interface in order to enable the master to receive the SCI control signals , if it is a read transaction the read data, send by the slave.

3.2.9 SRI Crossbar Registers

Figure 3-8 and **Table 3-8** are showing the address maps with all registers of the SRI Crossbar (XBar_SRI) module.

XBar_SRI Unit Register Overview

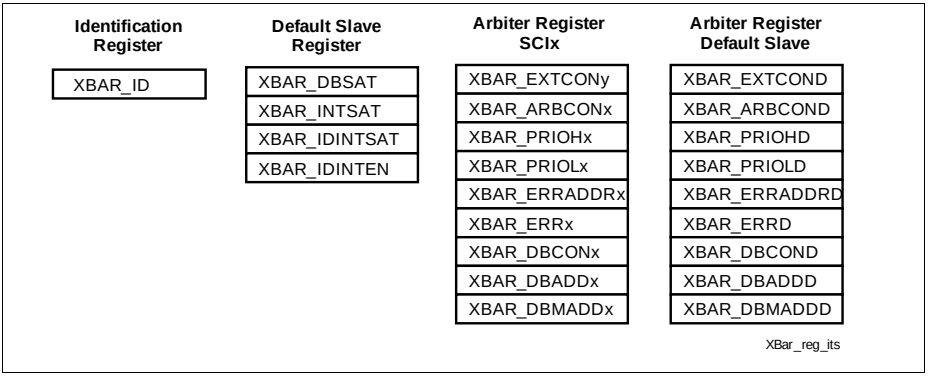


Figure 3-8 TC21x/TC22x/TC23x Control Registers

List of used Reset Class abbreviations:

- Reset Class 1 -> Debug Reset (see description in the chapter SCU / Reset Types)
- Reset Class 3 -> Power On Reset, Application Reset, System Reset (see description in the chapter SCU / Reset Types)

*Note: Addresses listed in column "Offset Address" of **Table 3-8** are word (32-bit) addresses.*

Note: XBar_SRI registers can be accessed only with SDTW (32 bit) transactions. 8, 16 bit and RMW transactions are not supported.

Table 3-8 Registers Address Space - XBar_SRI Register Address Space

Module	Base Address	End Address	Note
XBAR	F870 0000$_H$	F870 04FF$_H$	

Table 3-9 Registers Overview - Aurix_Bus Module Control Registers

Short Name	Description	Offset Addr.[1]	Access Mode		Reset Class	Description See
			Read	Write		
-	Reserved	400_H - 404_H	BE	BE	-	-
ID	Identification Register[2]	408_H	U, SV	BE	-	Page 3-34
DBSAT	Debug Trigger Event Status Register[2]	$40C_H$	U, SV	SV, P	1	Page 3-35
INTSAT	Arbiter Interrupt Status Register[2]	410_H	U, SV	SV, P	3	Page 3-36
IDINTSAT	ID Interrupt Status Register[2]	414_H	U, SV	SV, P	3	Page 3-39
IDINTEN	ID Interrupt Enable Register [2]	418_H	U, SV	SV, P	3	Page 3-41
-	Reserved	$41C_H$ - $4F4_H$	BE	BE	-	-
ACCEN1	Access Enable Register 1	$4F8_H$	U, SV	SV, SE	3	Page 3-64
ACCEN0	Access Enable Register 0	$4FC_H$	U, SV	SV, SE	3	Page 3-63
-	Reserved	$4FF_H$	BE	BE	-	-
EXTCOND	TC23x: External Slave Control (SFI control registers)	000_H	U, SV	SV, P	3	Page 3-43
-	TC23x: Reserved	000_H	BE	BE	-	-
ARBCOND	Arbiter Control Register Default Slave	004_H	U, SV	SV, P	3	Page 3-45
PRIOHD	Arbiter Priority Register High Default Slave	008_H	U, SV	SV, P	3	Page 3-47
PRIOLD	Arbiter Priority Register Low Default Slave	$00C_H$	U, SV	SV, P	3	Page 3-48
ERRADDRD	Arbiter Address Error/Debug Capture Register Default Slave	010_H	U, SV	SV, P	3	Page 3-50

Table 3-9 Registers Overview - Aurix_Bus Module Control Registers

Short Name	Description	Offset Addr.[1)]	Access Mode		Reset Class	Description See
			Read	Write		
ERRD	Arbiter Error/Debug Capture Register Default Slave	014_H	U, SV	SV, P	3	Page 3-53
DBCON	Arbiter x Debug Control Register	018_H	U, SV	SV, P	1	Page 3-56
DBADD	Arbiter x Debug Address Register	$01C_H$	U, SV	SV, P	1	Page 3-59
DBMADD	Arbiter x Debug Mask Address Register	020_H	U, SV	SV, P	1	Page 3-45
-	Reserved	024_H - 040_H	BE	BE	-	-
ARBCON0	Arbiter 0 Control Register	044_H	U, SV	SV, P	3	Page 3-45
PRIOH0	Arbiter 0 Priority Register High	048_H	U, SV	SV, P	3	Page 3-47
PRIOL0	Arbiter 0 Priority Register Low	$04C_H$	U, SV	SV, P	3	Page 3-48
ERRADDR0	Arbiter 0 Address Error/Debug Capture Register	050_H	U, SV	SV, P	3	Page 3-50
ERR0	Arbiter 0 Error/Debug Capture Register	054_H	U, SV	SV, P	3	Page 3-51
DBCON0	Arbiter 0 Debug Control Register	058_H	U, SV	SV, P	1	Page 3-53
DBADD0	Arbiter 0 Debug Address Register	$05C_H$	U, SV	SV, P	1	Page 3-56
DBMADD0	Arbiter 0 Debug Mask Address Register	060_H	U, SV	SV, P	1	Page 3-60
-	Reserved	064_H - 140_H	BE	BE	-	-
ARBCON4	TC23x: Arbiter 4 Control Register	144_H	U, SV	SV, P	3	Page 3-45
PRIOH4	TC23x: Arbiter 4 Priority Register High	148_H	U, SV	SV, P	3	Page 3-47

Table 3-9 Registers Overview - Aurix_Bus Module Control Registers

Short Name	Description	Offset Addr.[1]	Access Mode		Reset Class	Description See
			Read	Write		
PRIOL4	TC23x: Arbiter 4 Priority Register Low	$14C_H$	U, SV	SV, P	3	Page 3-48
ERRADDR4	TC23x: Arbiter 4 Address Error/Debug Capture Register	150_H	U, SV	SV, P	3	Page 3-50
ERR4	TC23x: Arbiter 4 Error/Debug Capture Register	154_H	U, SV	SV, P	3	Page 3-51
DBCON4	TC23x: Arbiter 4 Debug Control Register	158_H	U, SV	SV, P	1	Page 3-53
DBADD4	TC23x: Arbiter 4 Debug Address Register	$15C_H$	U, SV	SV, P	1	Page 3-57
DBMADD4	TC23x: Arbiter 4 Debug Mask Address Register	160_H	U, SV	SV, P	1	Page 3-61
-	TC22x: Reserved	144_H - 160_H	BE	BE	-	-
-	Reserved	164_H - $1C0_H$	BE	BE	-	-
ARBCON6	Arbiter 6 Control Register	$1C4_H$	U, SV	SV, P	3	Page 3-45
PRIOH6	Arbiter 6 Priority Register High	$1C8_H$	U, SV	SV, P	3	Page 3-47
PRIOL6	Arbiter 6 Priority Register Low	$1CC_H$	U, SV	SV, P	3	Page 3-47
ERRADDR6	Arbiter 6 Address Error/Debug Capture Register	$1D0_H$	U, SV	SV, P	3	Page 3-50
ERR6	Arbiter 6 Error/Debug Capture Register	$1D4_H$	U, SV	SV, P	3	Page 3-51
DBCON6	Arbiter 6 Debug Control Register	$1D8_H$	U, SV	SV, P	1	Page 3-53
DBADD6	Arbiter 6 Debug Address Register	$1DC_H$	U, SV	SV, P	1	Page 3-58

Table 3-9 Registers Overview - Aurix_Bus Module Control Registers

Short Name	Description	Offset Addr.[1]	Access Mode		Reset Class	Description See
			Read	Write		
DBMADD6	Arbiter 6 Debug Mask Address Register	$1E0_H$	U, SV	SV, P	1	Page 3-61
-	Reserved	$1E4_H$ - 200_H	BE	BE	-	-
ARBCON7	Arbiter 7 Control Register	204_H	U, SV	SV, P	3	Page 3-45
PRIOH7	Arbiter 7 Priority Register High	208_H	U, SV	SV, P	3	Page 3-47
PRIOL7	Arbiter 7 Priority Register Low	$20C_H$	U, SV	SV, P	3	Page 3-48
ERRADDR7	Arbiter 7 Address Error/Debug Capture Register	210_H	U, SV	SV, P	3	Page 3-50
ERR7	Arbiter 7 Error/Debug Capture Register	214_H	U, SV	SV, P	3	Page 3-51
DBCON7	Arbiter 7 Debug Control Register	218_H	U, SV	SV, P	1	Page 3-53
DBADD7	Arbiter 7 Debug Address Register	$21C_H$	U, SV	SV, P	1	Page 3-58
DBMADD7	Arbiter 7 Debug Mask Address Register	220_H	U, SV	SV, P	1	Page 3-62
-	Reserved	224_H - $3FF_H$	BE	BE	-	-

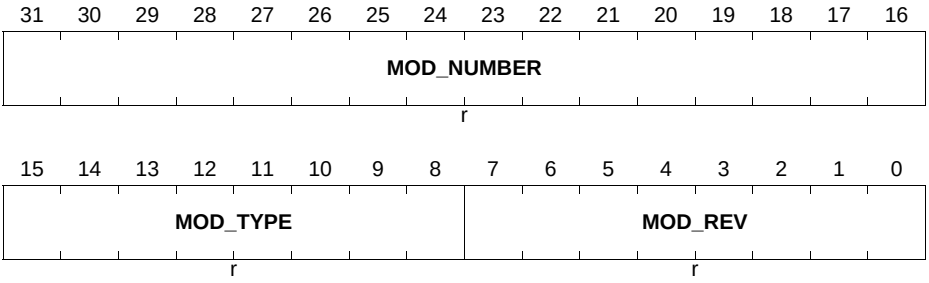
1) The absolute register address is calculated as follows:
Module Base Address (**Table 3-9**) + Offset Address (shown in this column)

2) This register is located inside the default slave

3.2.9.1 TC21x/TC22x/TC23x Control Registers

The identification register allows the programmer version-tracking of the module. The table below shows the identification register which is implemented in the LBCU module.

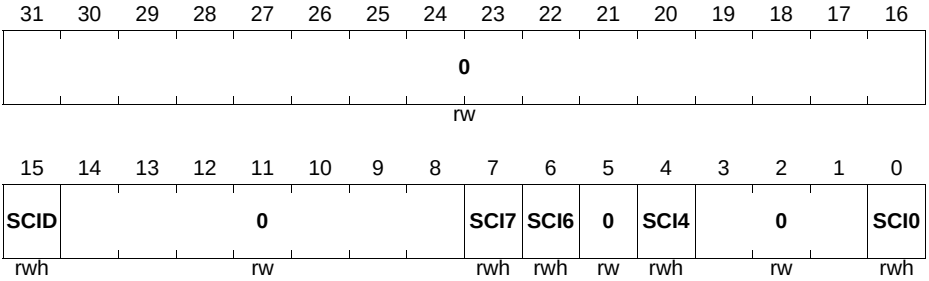
XBAR_ID
Module Identification Register (408$_H$) Reset Value: 0004 D0XX$_H$



Field	Bits	Type	Description
MOD_REV	[7:0]	r	**Module Revision Number** This bit field defines the module revision number. The value of a module revision starts with 01$_H$ (first revision).
MOD_TYPE	[15:8]	r	**Module Type** The bit field is set to C0$_H$ which defines the module as a 32-bit module.
MOD_NUMBER	[31:16]	r	**Module Number Value** This bit field defines a module identification number. The value for the LBCU module is 000F$_H$.

XBAR_DBSAT

Debug Trigger Event Status Register ($40C_H$) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
SCI0	0	rwh	**SCI Debug Trigger Event Status** 0_B No Debug Trigger Event was detected for SCI0 by its arbiter. 1_B A Debug Trigger Event was detected for SCI0 by its arbiter. Writing a '1' to this bit clears the bit.
SCI4	4	rwh	**SCI Debug Trigger Event Status** 0_B No Debug Trigger Event was detected for SCI4 by its arbiter. 1_B A Debug Trigger Event was detected for SCI4 by its arbiter. Writing a '1' to this bit clears the bit.
SCIn (n = 6-7)	n	rwh	**SCI Debug Trigger Event Status** 0_B No Debug Trigger Event was detected for SCIn by its arbiter. 1_B A Debug Trigger Event was detected for SCIn by its arbiter. Writing a '1' to this bit clears the bit.
SCID	15	rwh	**Default Slave Debug Trigger Event Status** 0_B No Debug Trigger Event was detected for the default slave by its arbiter. 1_B A Debug Trigger Event was detected for the default slave by its arbiter. Writing a '1' to this bit clears the bit.

Field	Bits	Type	Description
0	[31:16], [14:8], 5, [3:1]	rw	**Reserved** Read as 0; must be written with 0.

Note: This register is not reset with the normal system reset as all other registers in the XBar_SRI. This register is only reset with the special debug reset.

XBAR_INTSAT

Arbiter Interrupt Status Register (410$_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRSCID				**0**				**PRSCI7**	**PRSCI6**	**0**	**PRSCI4**		**0**		**PRSCI0**
rwh				r				rwh	rwh	r	rwh		r		rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCSCID				**0**				**SCSCI7**	**SCSCI6**	**0**	**SCSCI4**		**0**		**SCSCI0**
rwh				r				rwh	rwh	r	rwh		r		rwh

Field	Bits	Type	Description
SCSCI0	0	rwh	**Starvation Error from SCI0 Status** 0_B No starvation error is pending from SCI0 1_B A starvation error is pending from SCI0 Writing a zero to the bit leaves the content unchanged. Writing a one to the bit clears it. In case of a parallel clearing via software and an error from the hardware the bit remains set and is not cleared.

Field	Bits	Type	Description
SCSCI4	4	rwh	**Starvation Error from SCI4 Status** 0_B No starvation error is pending from SCI4 1_B A starvation error is pending from SCI4 Writing a zero to the bit leaves the content unchanged. Writing a one to the bit clears it. In case of a parallel clearing via software and an error from the hardware the bit remains set and is not cleared.
SCSCIn (n = 6-7)	n	rwh	**Starvation Error from SCIn Status** 0_B No starvation error is pending from SCIn 1_B A starvation error is pending from SCIn Writing a zero to the bit leaves the content unchanged. Writing a one to the bit clears it. In case of a parallel clearing via software and an error from the hardware the bit remains set and is not cleared.
SCSCID	15	rwh	**Starvation Error from Default Slave Status** 0_B No starvation error is pending from default slave 1_B A starvation error is pending from default slave Writing a zero to the bit leaves the content unchanged. Writing a one to the bit clears it. In case of a parallel clearing via software and an error from the hardware the bit remains set and is not cleared.
PRSCI0	16	rwh	**Protocol Error from SCI0 Status** 0_B No protocol error is pending from SCI0 1_B A protocol error is pending from SCI0 Writing a zero to the bit leaves the content unchanged. Writing a one to the bit clears it. In case of a parallel clearing via software and an error from the hardware the bit remains set and is not cleared.

Field	Bits	Type	Description
PRSCI4	20	rwh	**Protocol Error from SCI4 Status** 0_B No protocol error is pending from SCI4 1_B A protocol error is pending from SCI4 Writing a zero to the bit leaves the content unchanged. Writing a one to the bit clears it. In case of a parallel clearing via software and an error from the hardware the bit remains set and is not cleared.
PRSCIn (n = 6-7)	n+16	rwh	**Protocol Error from SCIn Status** 0_B No protocol error is pending from SCIn 1_B A protocol error is pending from SCIn Writing a zero to the bit leaves the content unchanged. Writing a one to the bit clears it. In case of a parallel clearing via software and an error from the hardware the bit remains set and is not cleared.
PRSCID	31	rwh	**Protocol Error from Default Slave Status** 0_B No protocol error is pending from default slave 1_B A protocol error is pending from default slave Writing a zero to the bit leaves the content unchanged. Writing a one to the bit clears it. In case of a parallel clearing via software and an error from the hardware the bit remains set and is not cleared.
0	[30:24], 21, [19:17], [14:8], 5, [3:1]	r	**Reserved** Read as 0; should be written with 0.

Note: Only the bits assigned to configured SCIs are implemented. Not implemented bits treated as reserved bits, read as ´0´, should be written with ´0´.

XBAR_IDINTSAT

Transaction ID Interrupt Status Register(414_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		IDMCI13	IDMCI12	0						IDMCI5	0				IDMCI0
r		rwh	rwh	r						rwh	r				rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDSCID	0							IDSCI7	IDSCI6	0	IDCSCI4	0			IDSCI0
rwh	r							rwh	rwh	r	rwh	r			rwh

Field	Bits	Type	Description
IDSCI0	0	rwh	**Transaction ID Error from SCI0 Status** 0_B No transaction ID error is pending from SCI0 1_B A transaction ID error is pending from SCI0 Writing a zero to the bit leaves the content unchanged. Writing a one to the bit clears it. *Note: In case of a parallel clearing via software and an error from the hardware the bit remains set and is not cleared.*
IDCSCI4	4	rwh	**Transaction ID Error from SCI4 Status** 0_B No transaction ID error is pending from SCI4 1_B A transaction ID error is pending from SCI4 Writing a zero to the bit leaves the content unchanged. Writing a one to the bit clears it. *Note: In case of a parallel clearing via software and an error from the hardware the bit remains set and is not cleared.*

Field	Bits	Type	Description
IDSCIn (n =6-7)	n	rwh	**Transaction ID Error from SCIn Status** 0_B No transaction ID error is pending from SCIn 1_B A transaction ID error is pending from SCIn Writing a zero to the bit leaves the content unchanged. Writing a one to the bit clears it. *Note: In case of a parallel clearing via software and an error from the hardware the bit remains set and is not cleared.*
IDSCID	15	rwh	**Transaction ID Error from Default Slave Status** 0_B No transaction ID error is pending from default slave 1_B A transaction ID error is pending from default slave Writing a zero to the bit leaves the content unchanged. Writing a one to the bit clears it. In case of a parallel clearing via software and an error from the hardware the bit remains set and is not cleared.
IDMCI0	16	rwh	**Transaction ID Error from MCI0 Status** 0_B No transaction ID error is pending from MCIn 1_B A transaction ID error is pending from MCIn Writing a zero to the bit leaves the content unchanged. Writing a one to the bit clears it. *Note: In case of a parallel clearing via software and an error from the hardware the bit remains set and is not cleared.*
IDMCI5	21	rwh	**Transaction ID Error from MCI5 Status** 0_B No transaction ID error is pending from MCIn 1_B A transaction ID error is pending from MCIn Writing a zero to the bit leaves the content unchanged. Writing a one to the bit clears it. *Note: In case of a parallel clearing via software and an error from the hardware the bit remains set and is not cleared.*

Field	Bits	Type	Description
IDMCIn (n = 12-13)	n+16	rwh	**Transaction ID Error from MCIn Status** 0_B No transaction ID error is pending from MCIn 1_B A transaction ID error is pending from MCIn Writing a zero to the bit leaves the content unchanged. Writing a one to the bit clears it. *Note: In case of a parallel clearing via software and an error from the hardware the bit remains set and is not cleared.*
0	[31:30], [27:22], [20:17], [14:8], 5, [3:1]	r	**Reserved** Read as 0; should be written with 0.

Note: Only the bits assigned to configured SCIs are implemented. Not implemented bits treated as reserved bits, read as ´0´, should be written with ´0´.

XBAR_IDINTEN

Transaction ID Interrupt Enable Register(418_H) **Reset Value: $3031\ 80D1_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		ENMCI13	ENMCI12	0						ENMCI5	0				ENMCI0
r		rw	rw	r						rw	r				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENSCID	0							ENSCI7	ENSCI6	0	ENSCI4	0			ENSCI0
rw	r							rw	rw	r	rw	r			rw

Field	Bits	Type	Description
ENSCI0	0	rw	**Enable ID Error from SCI0** 0_B No transaction ID error from SCI0 are sampled 1_B A transaction ID error from SCI0 are sampled

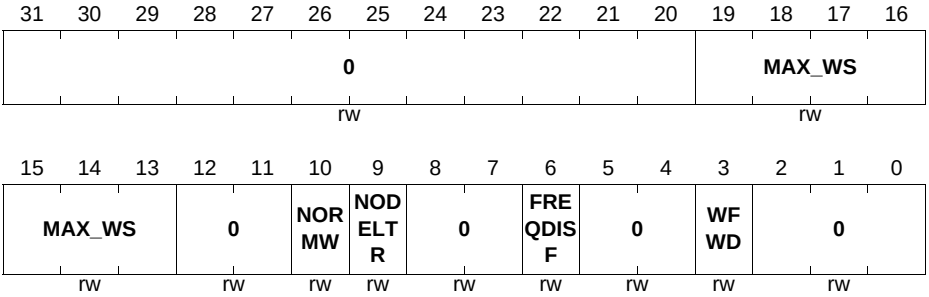
Field	Bits	Type	Description
ENSCI4	4	rw	**Enable ID Error from SCI4** 0_B No transaction ID error from SCI4 are sampled 1_B A transaction ID error from SCI4 are sampled
ENSCIn (n = 6-7)	n	rw	**Enable ID Error from SCIn** 0_B No transaction ID error from SCIn are sampled 1_B A transaction ID error from SCIn are sampled
ENSCID	15	rw	**Enable ID Error from Default Slave** 0_B No transaction ID error from the default slave is sampled 1_B A transaction ID error from the default slave is sampled
ENMCI0	16	rw	**Enable ID Error from MCI0** 0_B No transaction ID error from MCIn are sampled 1_B A transaction ID error from MCIn are sampled
ENMCI5	21	rw	**Enable ID Error from MCI5** 0_B No transaction ID error from MCIn are sampled 1_B A transaction ID error from MCIn are sampled
ENMCIn (n = 12-13)	n+16	rw	**Enable ID Error from MCIn** 0_B No transaction ID error from MCIn are sampled 1_B A transaction ID error from MCIn are sampled
0	[31:30], [27:22], [20:17], [14:8], 5, [3:1]	r	**Reserved** Read as 0; should be written with 0.

Note: Only the bits assigned to configured SCIs are implemented. Not implemented bits treated as reserved bits, read as ´0´, should be written with ´0´.

Note: Reset values for bits/bit fields coupled to masters or slaves that are not configured or enabled are zero.

XBAR_EXTCOND

External Control Register D **(000_H)** **Reset Value: 0000 0200$_H$**



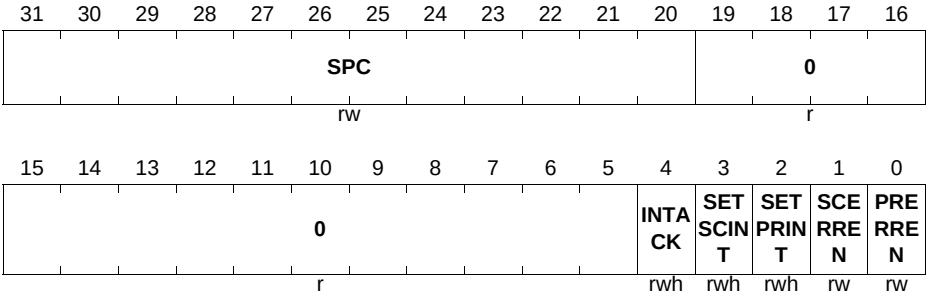
Field	Bits	Type	Description
WFWD	3	rw	**Wait for FPI Write Data** For FPI-Bus block write transfers the transaction request can be delayed until all write data arrived from the FPI-Bus in the SFI. As on thew FPI-Bus side very slow masters can resident SRI-Bus slaves can blocked for many SRI-Bus cycles if the write transaction is started with the first write data 0_B Write transactions on the SRI-Bus are requested with the first received write data from the FPI-Bus (default) 1_B Write transactions on the SRI-Bus are requested with the last received write data from the FPI-Bus
FREQDISF	6	rw	**Disable Fast Request Feature for FPI to SRI Transactions** 0_B Fast request feature is enabled (default) 1_B Fast request feature is disabled
NODELTR	9	rw	**Control Signal for deferred transactions** 0_B Deferred Transactions are generated 1_B Deferred Transactions are not generated (default)

Field	Bits	Type	Description
NORMW	10	rw	**Control Signal for deferred transactions** 0_B Deferred Transactions are generated for RMW (default) 1_B Deferred Transactions are not generated for RMW
MAX_WS	[19:13]	rw	**FPI-Bus Wait State Retry Ratio** SFI slave interface retries a read transaction to the FPI bus after the programmed value of wait states to change the read transaction in a delayed read transaction.
0	[31:20], [12:11], [8:7], [5:4], [2:0]	rw	**Reserved** Read as 0; shall be written with 0.

Note: Only the bits assigned to configured SCIs are implemented. Not implemented bits treated as reserved bits, read as ´0´, should be written with ´0´.

Note: Reset values for bits/bit fields coupled to masters or slaves that are not configured or enabled are zero.

XBAR_ARBCON0
Arbiter Control Register 0 **(044_H)** **Reset Value: FFF0 0003_H**
XBAR_ARBCON4
Arbiter Control Register 4 **(144_H)** **Reset Value: FFF0 0003_H**
XBAR_ARBCONx (x = 6-7)
Arbiter Control Register x **(044_H+x*40_H)** **Reset Value: FFF0 0003_H**
XBAR_ARBCOND
Arbiter Control Register D **(004_H)** **Reset Value: FFF0 0003_H**



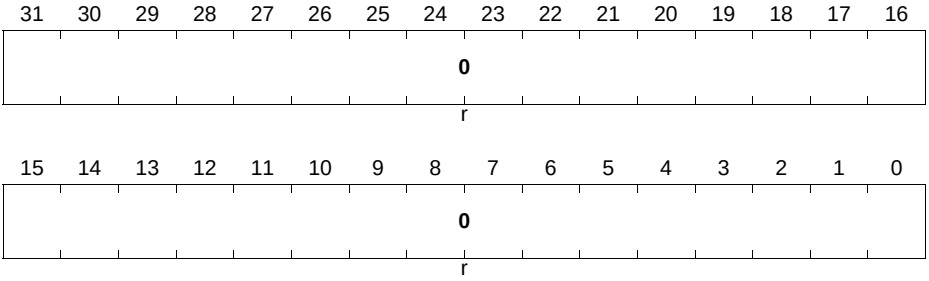
Field	Bits	Type	Description
PRERREN	0	rw	**SRI Protocol Error Enable** 0_B Protocol errors are not recognized and no information is captured. 1_B Protocol errors are recognized and information is captured.
SCERREN	1	rw	**SRI Starvation Error Enable** 0_B Starvation based errors are not recognized and no information is captured. 1_B Starvation based errors are recognized and information is captured.
SETPRINT	2	rwh	**Set SRI Protocol Interrupt** 0_B No protocol interrupt is generated 1_B A protocol interrupt is generated After the interrupt is generated by set the bit it's automatically cleared by the hardware

Field	Bits	Type	Description
SETSCINT	3	rwh	**Set SRI Starvation Interrupt** 0_B No starvation interrupt is generated 1_B A starvation interrupt is generated After the interrupt is generated by set the bit it's automatically cleared by the hardware
INTACK	4	rwh	**Interrupt Acknowledge** 0_B Default value 1_B An Error for this arbiter is pending. The ERRADDR and ERR registers are not updated for new errors. Writing a one to this bit field while it's set have the following results: The error lock of registers ERRADDR and ERR are released and the register could be updated with the next interrupt request detected (see **Chapter 3.2.7.5**). In the cycle after the write action the hardware automatically clears the bit.
SPC	[31:20]	rw	**Starvation Protection Counter Reload Value** The reload value defines the period for the starvation protection.
0	[19:5]	r	**Reserved** Read as 0; should be written with 0.

Note: Only the bits assigned to configured SCIs are implemented. Bits for non configured SCIs are treated as reserved bits.

Note: The ´D´ at ARBCOND stands for Default Slave.

XBAR_PRIOH0
Arbiter Priority Register 0 **(048_H)** **Reset Value: 0000 0000$_H$**
XBAR_PRIOH4
Arbiter Priority Register 4 **(148_H)** **Reset Value: 0055 5555$_H$**
XBAR_PRIOHx (x = 6-7)
Arbiter Priority Register x **(048_H+x*40_H)** **Reset Value: 0000 0000$_H$**
XBAR_PRIOHD
Arbiter Priority Register D **(008_H)** **Reset Value: 0000 0000$_H$**



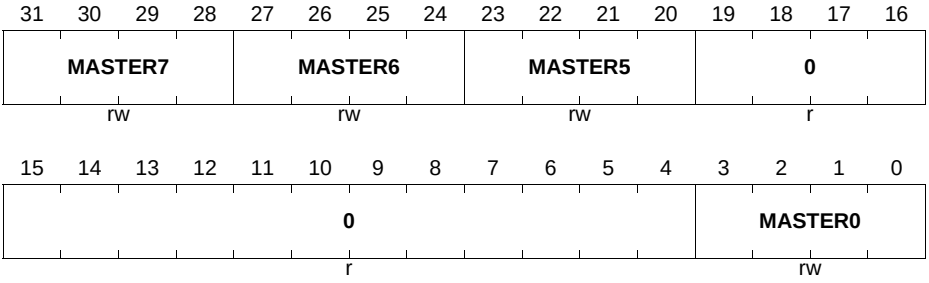
Field	Bits	Type	Description
0	[31:0]	r	**Reserved** Read as 0; should be written with 0.

Note: Only the bits assigned to configured MCIs are implemented. Bits for non configured SCIs are treated as reserved bits.

Note: The ´D´ at XBAR_PRIOLD stands for Default Slave.

Note: Reset values for bits/bit fields coupled to masters or slaves that are not configured or enabled are zero.

XBAR_PRIOLD
Arbiter Priority Register D **($00C_H$)** **Reset Value: $7654\ 0000_H$**
XBAR_PRIOL0
Arbiter Priority Register 0 **($04C_H$)** **Reset Value: $7654\ 0000_H$**
XBAR_PRIOL4
Arbiter Priority Register 4 **($14C_H$)** **Reset Value: $7654\ 0000_H$**
XBAR_PRIOLx (x = 6-7)
Arbiter Priority Register x **($04C_H$+x*40_H)** **Reset Value: $7654\ 0000_H$**



Field	Bits	Type	Description
MASTER0	[3:0]	rw	**Master 0 Priority** (Priority of DMA access) This bit field contains the master priority for the arbitration used by the arbiter of slave x. For each master a unique number for this slave has to be used. A lower number has a higher priority in the arbitration round than a higher one.
MASTER5	[23:20]	rw	**Master 5 Priority** (Priority of SFI access) This bit field contains the master priority for the arbitration used by the arbiter of slave x. For each master a unique number for this slave has to be used. A lower number has a higher priority in the arbitration round than a higher one.

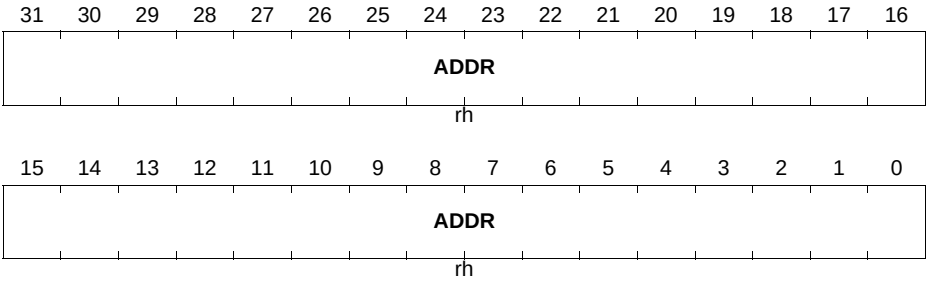
Field	Bits	Type	Description
MASTER6	[27:24]	rw	**Master 6 Priority** (Priority of CPU0.DMI access) This bit field contains the master priority for the arbitration used by the arbiter of slave x. For each master a unique number for this slave has to be used. A lower number has a higher priority in the arbitration round than a higher one.
MASTER7	[31:28]	rw	**Master 7 Priority** (Priority of CPU0.PMI access) This bit field contains the master priority for the arbitration used by the arbiter of slave x. For each master a unique number for this slave has to be used. A lower number has a higher priority in the arbitration round than a higher one.
0	[19:4]	r	**Reserved** Read as 0; should be written with 0.

Note: Only the bits assigned to configured MCIs are implemented. Bits for non configured SCIs are treated as reserved bits.

Note: The ´D´ at XBAR_PRIOLD stands for Default Slave.

Note: Reset values for bits/bit fields coupled to masters or slaves that are not configured or enabled are zero.

XBAR_ERRADDRD
Error/Debug Address Capture Register D(010_H) **Reset Value: $7000\ 0000_H$**
XBAR_ERRADDR0
Error/Debug Address Capture Register 0(050_H) **Reset Value: $6000\ 0000_H$**
XBAR_ERRADDR4
Error/Debug Address Capture Register 4(150_H) **Reset Value: $8000\ 0000_H$**
XBAR_ERRADDR6
Error/Debug Address Capture Register 6($1D0_H$) **Reset Value: $8800\ 0000_H$**
XBAR_ERRADDR7
Error/Debug Address Capture Register 7(210_H) **Reset Value: $8020\ 0000_H$**

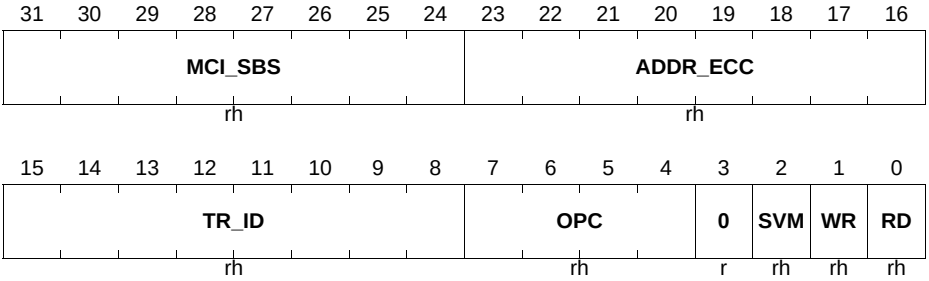


Field	Bits	Type	Description
ADDR	[31:0]	rh	**Transaction Address** This biffield contains the address of the erroneous transaction from the address phase

Note: The default value can differ from the one shown here because a constant can be used to reduce the number of compared bits in the arbitration if a slave occupies only a limited address area. For more details see the design specification of the XBar_SRI.

Note: The ´D´ at XBAR_ERRADDRD stands for Default Slave.

XBAR_ERR0
Error/Debug Capture Register 0 (054_H) Reset Value: 0000 0000$_H$
XBAR_ERR4
Error/Debug Capture Register 4 (154_H) Reset Value: 0000 0000$_H$
XBAR_ERRx (x = 6-7)
Error/Debug Capture Register x (054_H+x*40_H) Reset Value: 0000 0000$_H$
XBAR_ERRD
Error/Debug Capture Register D (014_H) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
RD	0	rh	**Read Indication Status** 0_B The read indication SRI-Bus signal line was asserted (read or start of RMW transaction) 1_B The read indication SRI-Bus signal line was deasserted (no read transaction)
WR	1	rh	**Write Indication Status** 0_B The write indication SRI-Bus signal line was asserted (write or start of RMW transaction) 1_B The write indication SRI-Bus signal line was deasserted (no write transaction)
SVM	2	rh	**Supervisor Mode Indication Status** 0_B The supervisor mode indication SRI-Bus signal line was deasserted 1_B The supervisor mode indication SRI-Bus signal line was asserted
OPC	[7:4]	rh	**Operation Code** This bit field contains the op-code of the erroneous transaction.

Field	Bits	Type	Description
TR_ID	[15:8]	rh	**Transaction ID** This bit field contains the transaction ID of the erroneous transaction from the address phase. The Transaction ID is build out of an 6 bit unique TAG ID TR_ID[5:0] and a 2 bit running number TR_ID[7:6] (see also **Chapter 3.6**).
ADDR_ECC	[23:16]	rh	**SRI Address Phase ECC** This bit field contains the Address Phase ECC of the erroneous transaction *Note: In the current implementation the ECC code is only used for error detection. Detected errors are reported to the SMU but not corrected.*
MCI_SBS	[31:24]	rh	**MCI Sideband Signals [7:0]** This bit field contains the MCI Sideband Signals [7:0] that are related to the address phase informations captured by the ERRD/ERRADDR registers. In the TC21x/TC22x/TC23x Family the sideband signals are used by the DMA SRI master interface to provide information about the DMA requestor of a DMA transaction (for the encoding see **Table 3-5**).
0	3	r	**Reserved** Read as 0; should be written with 0

Note: The ´D´ at XBAR_ERRD stands for Default Slave.

XBAR_DBCON0
Debug Control Register 0 **(058_H)** **Reset Value: 0000 0000$_H$**
XBAR_DBCON4
Debug Control Register 4 **(158_H)** **Reset Value: 0000 0000$_H$**
XBAR_DBCONx (x = 6-7)
Debug Control Register x **(058_H+x*40_H)** **Reset Value: 0000 0000$_H$**
XBAR_DBCOND
Debug Control Register D **(018_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				MASTER				MASEN	0		ERREN	ADDEN	SVMEN	WREN	RDEN
r				rw				rw	r		rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												SETDBEVT	REARM	DBSAT	DBEN
r												w	w	rh	r

Field	Bits	Type	Description
DBEN	0	r	**Status of OCDS Enable Signal** Displays the value of the OCDS enable signal from Cerberus.
DBSAT	1	rh	**Debug (OCDS) Trigger Status** 0_B The debug (OCDS) trigger was used and has to be rearmed 1_B The debug (OCDS) trigger is armed

Field	Bits	Type	Description
REARM	2	w	**Rearm Debug (OCDS) Trigger** 0_B Read back value 1_B Writing a one to this bit arms sets bit DBCON.DBSAT. Writing a one to this bit field while it's set have the following results: The debug lock of registers ERRADDR and ERR are released and the register could be updated with the next debug event request detected (see **Chapter 3.2.7.5**). In the cycle after the write action the hardware automatically clears the bit. *Note: This bit is automatically reset by the hardware after DBCON.DBSAT was set.*
SETDBEVT	3	w	**Set Debug Event** 0_B Default value 1_B A debug trigger event will be generated by this arbiter if the debug feature is enabled (DBCON.ENST is set). The registers ERR and ERRADD capture the status if not locked already. *Note: This bit is automatically reset by the hardware.*
RDEN	16	rw	**Read Trigger Enable** 0_B Read Transaction are not used to trigger the debug trigger event (OCDS) 1_B Read Transaction are used to trigger the debug trigger event (OCDS)
WREN	17	rw	**Write Trigger Enable** 0_B Write Transaction are not used to trigger the debug trigger event (OCDS) 1_B Write Transaction are used to trigger the debug trigger event (OCDS)
SVMEN	18	rw	**SVM Trigger Enable** 0_B SVM Transaction are not used to trigger the debug trigger event (OCDS) 1_B SVM Transaction are used to trigger the debug trigger event (OCDS)

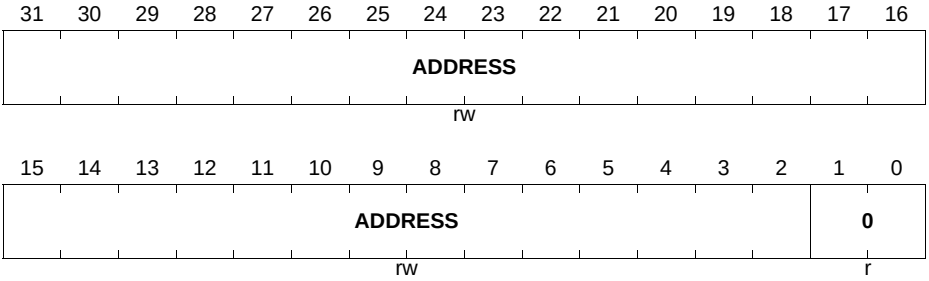
Field	Bits	Type	Description
ADDEN	19	rw	**Address Trigger Enable** 0_B Transaction addresses are not used to trigger the debug trigger event (OCDS) 1_B Transaction address defined by the registers DBADD and DBMADD are used to trigger the debug trigger event (OCDS)
ERREN	20	rw	**Error Trigger Enable** 0_B Errored Transactions are not used to trigger the debug trigger event (OCDS) 1_B Errored Transactions are used to trigger the debug trigger event (OCDS) Reading this bit return always zero. *Note: Protocol errors, starvation errors and transaction ID errors can be used where, but have to enabled before as usual in registers ARBCON or IDINTEN depending on the error type.*
MASEN	23	rw	**Master Trigger Enable** 0_B The Master TAG ID as defined in the bit field MASTER is not used to trigger the debug trigger event (OCDS) 1_B The Master TAG ID as defined in the bit field MASTER is used to trigger the debug trigger event (OCDS)
MASTER	[29:24]	rw	**Master TAG ID Trigger Selector** The value of this bit field define the Master TAG ID within the address phase of an SRI transaction to the related SRI slave module that triggers the debug trigger event (OCDS). The Master TAG IDs are defined here: **Table 3-15**.
0	[31:30], [22:21], [15:4]	r	**Reserved** Read as 0; should be written with 0.

Note: This register is not reset with the normal system reset as all other registers in the XBar_SRI. This register is only reset with the special debug reset.

Note: The ´D´ at XBAR_DBCOND stands for Default Slave.

XBAR_DBADDD

Debug Address Register D **($01C_H$)** **Reset Value: 0000 0000_H**



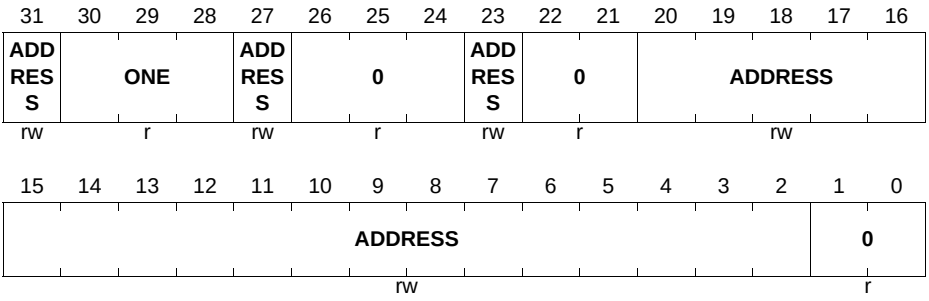
Field	Bits	Type	Description
ADDRESS	[31:2]	rw	**Debug Address Boundary**
0	[1:0]	r	**Reserved** Read as 0; should be written with 0.

Note: This register is reset with the debug reset (Class 1 reset).

Note: The last ´D´ at XBAR_DBADDD stands for Default Slave.

XBAR_DBADD0

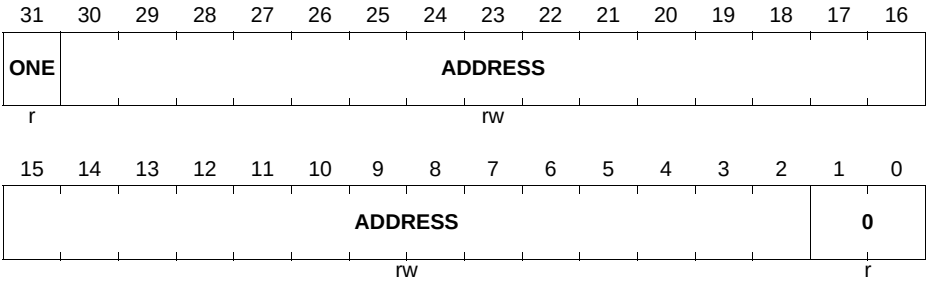
Debug Address Register 0 **($05C_H$)** **Reset Value: 7000 0000_H**



Field	Bits	Type	Description
ADDRESS, ADDRESS, ADDRESS, ADDRESS	27, 23, [20:2], 31	rw	**Debug Address Boundary**
ONE	[30:28]	r	**Reserved** Read as 1; should be written with 0.
0	[26:24], [22:21], [1:0]	r	**Reserved** Read as 0; should be written with 0.

Note: This register is reset with the debug reset (Class 1 reset).

XBAR_DBADD4
Debug Address Register 4 ($15C_H$) Reset Value: $8000\ 0000_H$



Field	Bits	Type	Description
ADDRESS	[30:2]	rw	**Debug Address Boundary**
ONE	31	r	**Reserved** Read as 1; should be written with 0.
0	[1:0]	r	**Reserved** Read as 0; should be written with 0.

Note: This register is reset with the debug reset (Class 1 reset).

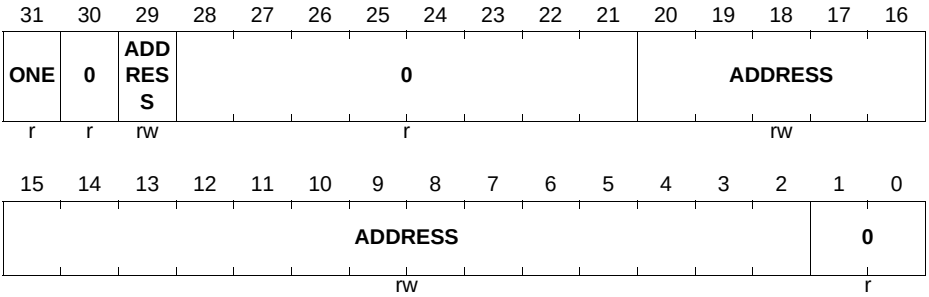
XBAR_DBADD6
Debug Address Register 6 ($1DC_H$) Reset Value: $8000\ 0000_H$



Field	Bits	Type	Description
ADDRESS, ADDRESS	[30:28], [26:2]	rw	**Debug Address Boundary**
ONE	31	r	**Reserved** Read as 1; should be written with 0.
0	27, [1:0]	r	**Reserved** Read as 0; should be written with 0.

Note: This register is reset with the debug reset (Class 1 reset).

XBAR_DBADD7
Debug Address Register 7 ($21C_H$) Reset Value: $8800\ 0000_H$



Field	Bits	Type	Description
ADDRESS, ADDRESS	29, [20:2]	rw	**Debug Address Boundary**
ONE	31	r	**Reserved** Read as 1; should be written with 0.
0	30, [28:21], [1:0]	r	**Reserved** Read as 0; should be written with 0.

Note: This register is reset with the debug reset (Class 1 reset).

XBAR_DBMADDD
Debug Mask Address Register D (020_H) Reset Value: $0000\ 0000_H$



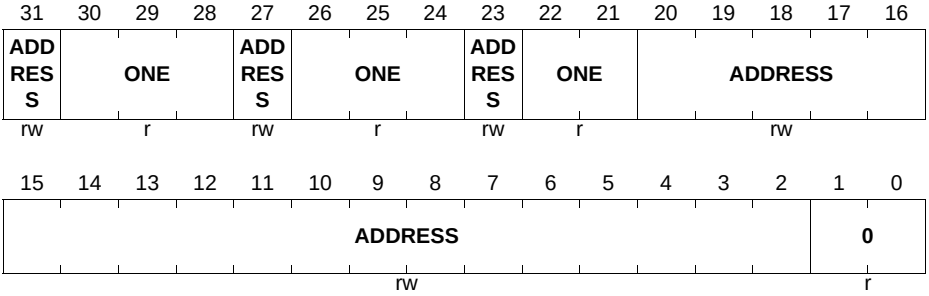
Field	Bits	Type	Description
ADDRESS	[31:0]	rw	**Debug Address Boundary**

Note: This register is reset with the debug reset (Class 1 reset).

Note: The last ´D´ at XBAR_DBMADDD stands for Default Slave.

XBAR_DBMADD0

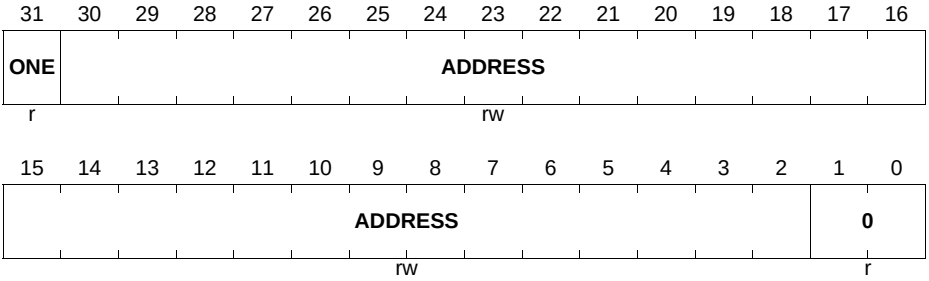
Debug Mask Address Register 0 (060_H) Reset Value: 7760 0000$_H$



Field	Bits	Type	Description
ADDRESS, ADDRESS, ADDRESS, ADDRESS	[20:2], 27, 23, 31	rw	**Debug Address Boundary**
ONE, ONE, ONE	[30:28], [26:24], [22:21]	r	**Reserved** Read as 1; should be written with 0.
0	[1:0]	r	**Reserved** Read as 0; should be written with 0.

Note: This register is reset with the debug reset (Class 1 reset).

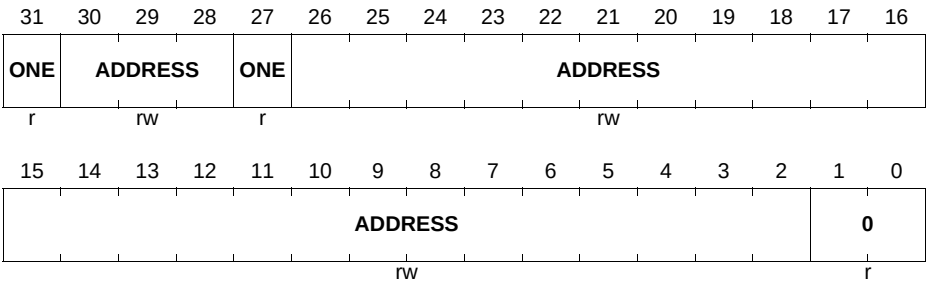
XBAR_DBMADD4
Debug Mask Address Register 4 (160_H) Reset Value: $8000\ 0000_H$



Field	Bits	Type	Description
ADDRESS	[30:2]	rw	**Debug Address Boundary**
ONE	31	r	**Reserved** Read as 1; should be written with 0.
0	[1:0]	r	**Reserved** Read as 0; should be written with 0.

Note: This register is reset with the debug reset (Class 1 reset).

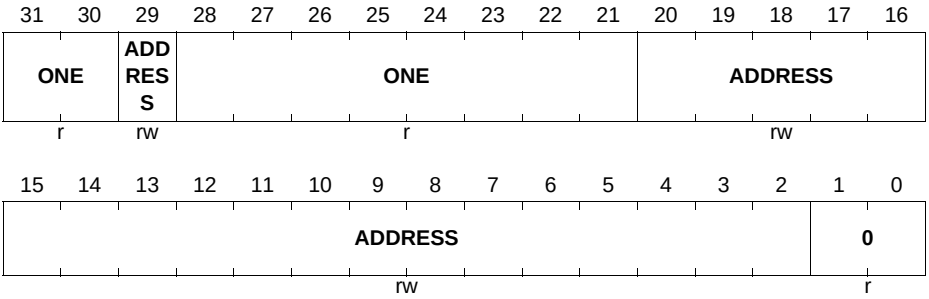
XBAR_DBMADD6
Debug Mask Address Register 6 ($1E0_H$) Reset Value: $8800\ 0000_H$



Field	Bits	Type	Description
ADDRESS, ADDRESS	[30:28], [26:2]	rw	**Debug Address Boundary**
ONE, ONE	31, 27	r	**Reserved** Read as 1; should be written with 0.
0	[1:0]	r	**Reserved** Read as 0; should be written with 0.

Note: This register is reset with the debug reset (Class 1 reset).

XBAR_DBMADD7
Debug Mask Address Register 7 (220_H) Reset Value: DFE0 0000_H



Field	Bits	Type	Description
ADDRESS, ADDRESS	29, [20:2]	rw	**Debug Address Boundary**
ONE, ONE	[31:30], [28:21]	r	**Reserved** Read as 1; should be written with 0.
0	[1:0]	r	**Reserved** Read as 0; should be written with 0.

Note: This register is reset with the debug reset (Class 1 reset).

Access Enable Register 0 (ACCEN0)

The Access Enable Register 0 controls write access for transactions with the on chip bus master TAG ID 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000_B, EN1 -> TAG ID 000001_B , ... ,EN31 -> TAG ID 011111_B.

XBAR_ACCEN0
Access Enable Register 0 **($4FC_H$)** **Reset Value: $FFFF\ FFFF_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

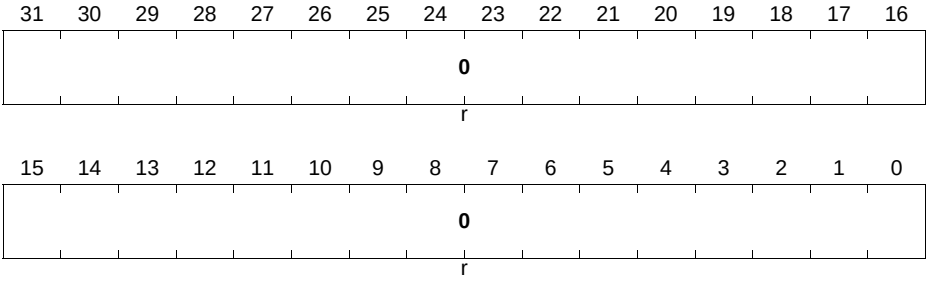
Field	Bits	Type	Description
ENn (n = 0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Access Enable Register 1 (ACCEN1)

The Access Enable Register 1 controls write access for transactions with the on chip bus master TAG ID 100000_B to 111111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the AURIX devices.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001B, ... ,EN31 -> TAG ID 111111B.

XBAR_ACCEN1
Access Enable Register 1 **($4F8_H$)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
0	[31:0]	r	**Reserved** Read as 0; should be written with 0.

3.3 Shared Resource Interconnect to FPI Bus Interface (SFI Bridge)

This section describes the basic functionality of the SFI Bridge.

3.3.1 Functional Overview

The SFI Bridge is implemented as an uni-directional bus bridge that forwards transactions from the System Peripheral Bus (SPB) to the SRI Interconnect. The bridge supports all transactions types of both the SRI Bus and FPI Bus.

The bridge is transparent, this means that the address of a transaction and the master TAG of a bus master is forwarded to the other side of the bridge. Addresses are only changed by the bridge where it is required by the transaction conversion from the 64 bit SRI Interconnect to the 32 bit SPB.

Bus Errors at Writes via the SFI Bridge

Write transactions are handled as posted writes. The SFI is able to buffer multiple posted writes. This means that a write operation from the SPB through the SFI Bridge to the SRI Interconnect can be finished on SPB and than generated on the SRI autonomously by the SFI. If this write operation results in a bus error on the SRI the Error information is not passed back to the SPB bus. This is also valid for transactions from SRI to SPB via SFI bridge. The bus error is detected by the on chip bus control logic of the target on chip bus system (BCU_FPI on the SPB, XBar_SRI on the SRI) which can generate an interrupt.

Note that this behavior occurs only at write operations via the SFI Bridge. It can also be triggered by an erroneous write cycle of a read-modify-write bus transaction.

3.4 System Peripheral Bus

The TC21x/TC22x/TC23x has one on-chip FPI Bus:

- System Peripheral Bus (SPB)
  - System bus for on-chip peripherals

This section gives an overview of the on-chip FPI Bus. It describes its bus control units, the bus characteristics, bus arbitration, scheduling, prioritizing, error conditions, and debugging support.

3.4.1 Overview

The FPI Bus interconnects the on-chip peripheral functional units with the TC21x/TC22x/TC23x processor subsystem.

The FPI Bus is designed to be quick to be acquired by on-chip functional units, and quick to transfer data. The low setup overhead of the FPI Bus access protocol guarantees fast FPI Bus acquisition, which is required for time-critical applications.

The FPI Bus is designed to sustain high transfer rates. For example, a peak transfer rate of up to 320 Mbyte/s can be achieved with the 32-bit data bus at 80 MHz bus clock. Multiple data transfers per bus arbitration cycle allow the FPI Bus to operate close to its peak bandwidth.

Additional features of the FPI Bus include:

- Optimized for high speed and high performance
- Support of multiple bus masters and pipelined transactions
- 32-bit wide address and data buses
- 8-, 16-, and 32-bit data transfers
- 64-, 128-, and 256-bit block transfers
- Central simple per-cycle arbitration
- Slave-controlled wait state insertion
- Support of atomic operations LDMST, ST.T and SWAP.W
- Starvation prevention mechanism that can take care that even low priority requests will be granted after a configurable number of arbitration cycles, permanently enabled
- Default slave that takes over transactions no other slave responds
- Timout detection and handling
- Capturing of transaction information in case of a bus error that cn be released by again by SW incl. transaction address, control incl. op-code, data
- Address Phase includes Supervisor Mode information
- All SPB (FPI) slave modules implemented with a TAG ID based access protection that provides a generic write protection for the control registers

The functional units of the TC21x/TC22x/TC23x are connected to the FPI Bus via FPI Bus interfaces. An FPI Bus interfaces acts as bus agents, requesting bus transactions on behalf of their functional unit, or responding to bus transaction requests.

There are two types of bus agents:

- FPI Bus master agents can initiate FPI Bus transactions and can also act as slaves.
- Slave agents can only react and respond to FPI Bus transaction requests in order to read or write internal registers of slave modules as for example memories.

When an FPI Bus master attempts to initiate a transfer on the FPI Bus, it first signals a request for bus ownership to the bus control unit (SBCU). When bus ownership is granted by the SBCU, an FPI Bus read or write transaction is initiated. The unit targeted by the transaction becomes the FPI Bus slave, and responds with the requested action.

Some functional units operate only as slaves, while others can operate as either masters or slaves on the FPI Bus.

FPI Bus arbitration is performed by the Bus Control Unit (SBCU) of the FPI Bus. In case of bus errors, the SBCU generates an interrupt request to the CPU and provides debugging information about the actual bus error to the CPU.

3.4.2 Bus Transaction Types

This section describes the SPB transaction types.

Single Transfers

Single transfers are byte, half-word, and word transactions that target any slave connected to SPB. Note that the SFI Bridge operates as an SPB master.

Block Transfers

Block transfers operate in principle in the same way as single transfers do, but one address phase is followed by multiple data phases. Block transfers can be composed of 2 word, 4 word, or 8 word transfers.

Note: In general, block transfers (2 word, 4 word, or 8 word) cannot be executed in the TC21x/TC22x/TC23x with peripheral units that operate as FPI Bus slaves during an FPI Bus transaction.

Block transfers are initiated by the following CPU instructions: LD.D, LD.DA, MOV.D, ST.D and ST.DA. Additionally there are communication peripherals that are able to generate block transfers (e.g. Ethernet).

Atomic Transfers

Atomic transfers are generated by LDMST, ST.T and SWAP.W instructions that require two single transfers. The read and write transfer of an atomic transfer are always locked and cannot be interrupted by another bus masters. Atomic transfers are also referenced as read-modify-write transfers.

*Note: See also **Table 3-11** for available FPI Bus transfer types.*

3.4.3 Reaction of a Busy Slave

If an FPI Bus slave is busy at an incoming FPI Bus transaction request, it can delay the execution of the FPI Bus transaction. The requesting FPI Bus master releases the FPI Bus for one cycle after the FPI Bus transaction request, in order to allow the FPI Bus slave to indicate if it is ready to handle the requested FPI Bus transaction. This sequence is repeated as long as the slave indicates that it is busy.

Note: For the FPI Bus default master, the one cycle gap does not result in a performance loss because it is granted the FPI Bus in this cycle as default master if no other master requests the FPI Bus for some other reasons.

3.4.4 Address Alignment Rules

FPI Bus address generation is compliant with the following rules:

- Half-word transactions must have a half-word aligned address (A0 = 0). Half-word accesses on byte lanes 1 and 2 addresses are illegal.
- Word transactions must always have word-aligned addresses (A[1:0] = 00_B).
- Block transactions must always have block-type aligned addresses.

3.4.5 FPI Bus Basic Operations

This section describes some basic transactions on the FPI Bus.

The example in **Figure 3-9** shows the three cycles of an FPI Bus operation:

1. **Request/Grant Cycle:** The FPI Bus master attempts to perform a read or write transfer and requests for the FPI Bus. If the FPI Bus is available, it is granted in the same cycle by the FPI Bus controller.
2. **Address Cycle:** After the request/grant cycle, the master puts the address on the FPI Bus, and all FPI Bus slave devices check whether they are addressed for the following data cycle.
3. **Data Cycle:** In the data cycle, either the master puts write data on the FPI Bus which is read by the FPI Bus slave (write cycle) or vice versa (read cycle).

Transfers 2 and 3 show the conflict when two master try to use the FPI Bus and how the conflict is resolved. In the example, the FPI Bus master of transfer 2 has a higher priority than the FPI Bus master of transfer 3.

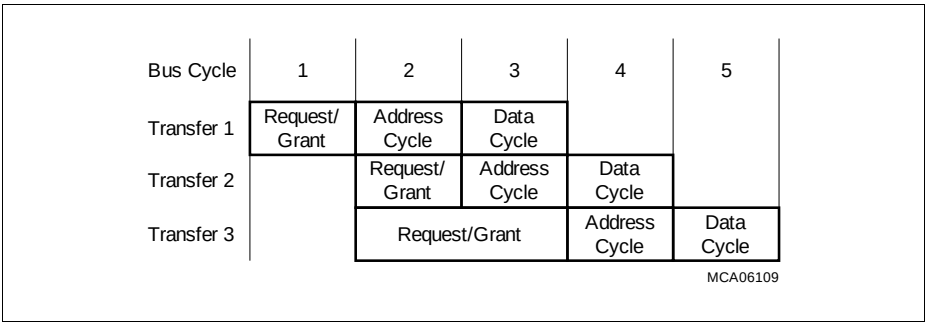


Figure 3-9 Basic FPI Bus Transactions

At a block transfer, the address cycle of a second transfer is extended until the data cycles of the block transfer are finished. In the example of **Figure 3-10**, transfer 1 is a block transfer, while transfer 2 is a single transfer.

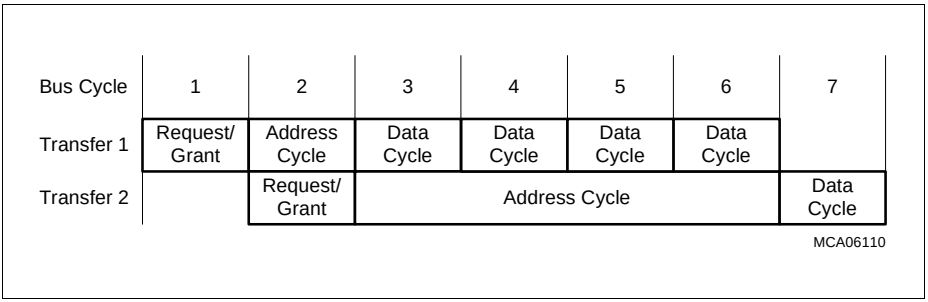


Figure 3-10 FPI Bus Block Transactions

3.5 FPI Bus Control Unit (SBCU)

The TC21x/TC22x/TC23x incorporates one FPI Bus control Unit (BCU) for the FPI based System Peripheral Bus (SPB), called System Bus Control Unit (SBCU).

3.5.1 FPI Bus Arbitration

The arbitration unit of the BCU determines whether it is necessary to arbitrate for FPI Bus ownership, and, if so, which available bus requestor gets the FPI Bus for the next data transfer. During arbitration, the bus is granted to the requesting agent with the highest priority. If no request is pending, the bus is granted to a default master. If no bus master takes the bus, the BCU itself will set the FPI Bus into an idle state.

3.5.1.1 Arbitration on the System Peripheral Bus

The TC21x/TC22x/TC23x SPB has multiple bus agents that can become SPB master:

- Each agent is assigned to 4-bit priority bit field in the PRIOH or in the PRIOL register
- The value in a priority bit field defines the related SPB master agent priority
- A lower priority number has a higher priority, ’0’ is the highest priority
- Each priority bit field in the PRIOH and PRIOL has a unique default value that can be changed during ramp up

Note: The DMA controller as an bus agent supports three priorities as this module covers two DMA Move Engines but also the Cerberus (JTAG/L1 Debug Interface) which might be used with the highest or lowest priority. Additionally the DMA Move Engine channels can be assigned to one of the three priorities.

3.5.1.2 Default Master

If no request is active, the FPI-Bus will be granted to a Default Master. The FPI master that was most recently granted will be granted to the Default Master. After reset, the master with the priority 0 will be the Default Master.

3.5.1.3 Arbitration Algorithms

The arbitration algorithm implemented in the BCU is based on:

- Priority driven arbitration of master agents with a pending request
- Starvation Prevention mechanism can increase master agent priorities during Starvation Prevention process

The default priority of each connected FPI master agent can be changed during runtime via the arbiter priority registers (PRIOH, PRIOL, see ´Changing Master Priorities´). There is a 4-bit priority bit field for each master agent in the PRIOH/PRIOL registers that defines the priority of this for this arbiter. After reset, each priority register has a default value, which means that each FPI master has unique default priority.

The default priority settings should be optimal for most applications. Where required, the arbitration setting can be adapted to the specific application needs (see ´Changing Master Priorities).

It must be ensured that two FPI master agents don’t have the same priority.

Priority Driven Arbitration

The general arbitration algorithm is priority driven where priority 0 is the highest priority and 15 the lowest one. If multiple masters are requesting for one slave, the master with the highest priority will win the next arbitration round (see also ’starvation prevention’).

Changing Master Priorities

Master priorities must be configured during ramp up as long only one FPI master agent is active (CPU0). Master priorities must not be changed while multiple FPI master agents are active / enabled.

3.5.1.4 Starvation Prevention

Starvation prevention is a feature of the SBCU that can take care that even requesting low priority master agents will be granted after a period, where the period length can be controlled by SBCU control registers. Because the priority assignment of the SPB agents is fixed, it is possible that a lower-priority bus requestor may never be granted the bus if a higher-priority bus requestor continuously asks for, and receives, bus ownership. To protect against bus starvation of lower-priority masters, the starvation prevention mechanism of the SBCU will detect such cases and momentarily raise the priority of the lower-priority requestor to the highest priority (above all other priorities), thereby guaranteeing it access.

Starvation protection employs a counter that is decremented each time an arbitration is performed on the connected FPI bus. The counter is re-loaded with the starvation period value in the SBCU_CON.SPC bit field as long it is enabled SBCU_CON.SPE. When this counter is counted down to zero, for each active bus request a request flag is stored in the BCU. This flag is cleared automatically when a master is granted the bus.

When the next period is finished, an active request of a master from which the request flag was set, a starvation event happened. This master will now be set to the highest priority and will be granted service. If there are several masters to which this starvation condition applies, they are served in the order of their configured priority ranking.

If a master that is processing its transaction under starvation condition is retried, its corresponding request flag is automatically again.

Starvation protection is permanently enabled. The sample period of the counter is programmed through bit field SBCU_CON.SPC. SPC should be set to a value at least greater than or equal to the number of masters. Its reset value is FF_H.

3.5.2 FPI Bus Error Handling

When an error occurs on an FPI Bus, its BCU captures and stores data about the erroneous condition and generates a service request if enabled to do so. The error conditions that force an error-capture are:

- Error Acknowledge: An FPI Bus slave responds with an error to a transaction.
- Un-implemented Address: No FPI Bus slave responds to a transaction request.
- Time-out: A slave does not respond to a transaction request within a certain time window. The number of bus clock cycles that can elapse until a bus time-out is generated is defined by bit field SBCU_CON.TOUT.

When a transaction causes an error, the address and data phase signals of the transaction causing the error are captured and stored in registers.

- The Error Address Capture Register (SBCU_EADD) stores the 32-bit FPI Bus address that has been captured during the erroneous FPI Bus transaction.
- The Error Data Capture Registers (SBCU_EDAT) stores the 32-bit FPI Bus data bus information that has been captured during the erroneous FPI Bus transaction.

- The Error Control Capture Register (SBCU_ECON) stores status information of the bus error event.

If more than one FPI Bus transaction generates a bus error, only the first bus error is captured. After a bus error has been captured, the capture mechanism must be released again by software. The lock is removed by reading the register SBCU_ECON which clears the SBCU_ECON.ERRCNT bit field.

Note: It is recommended to read in a debug season register ECON last as this removes the lock and a new error can already modify the content of the other two registers EDAT and EADD.

If a write transaction from TriCore causes an error on the SPB, the originating master is not informed about this error as it is not an SPB master agent. With each bus error-capture event, a service request is generated, and an interrupt can be generated if enabled and configured in the corresponding service request register.

Interpreting the BCU Control Register Error Information

Although the address and data values captured in registers SBCU_EADD and SBCU_EDAT, respectively, are self-explanatory, the captured FPI Bus control information needs some more explanation.

Register SBCU_ECON captures the state of the read (RDN), write (WRN), Supervisor Mode (SVM), acknowledge (ACK), ready (RDY), abort (ABT), time-out (TOUT), bus master identification lines (TAG) and transaction operation code (OPC) lines of the FPI Bus.

The SVM signal is set to 1 for an access in Supervisor Mode and set to 0 for an access in User Mode. The time-out signal indicates if there was no response on the bus to an access, and the programmed time (via SBCU_TOUT) has elapsed. TOUT is set to 1 in this case. An acknowledge code has to be driven by the selected slave during each data cycle of an access. These codes are listed in **Table 3-10**.

Table 3-10 FPI Bus Acknowledge Codes

Code (ACK)	Description
00_B	NSC: No Special Condition.
01_B	SPT: Split Transaction (not used in the TC21x/TC22x/TC23x).
10_B	RTY: Retry. Slave can currently not respond to the access. Master needs to repeat the access later.
11_B	ERR: Bus Error, last data cycle is aborted.

Transactions on the FPI Bus are classified via a 4-bit operation code (see **Table 3-11**). Note that split transactions (OPC = 1000_B to 1110_B) are not used in the TC21x/TC22x/TC23x.

Table 3-11 FPI Bus Operation Codes (OPC)

OPC	Description
0000_B	Single Byte Transfer (8-bit)
0001_B	Single Half-Word Transfer (16-bit)
0010_B	Single Word Transfer (32-bit)
0100_B	2-Word Block Transfer
0101_B	4-Word Block Transfer
0110_B	8-Word Block Transfer
1111	No operation
0011_B, 0111_B, 1000_B - 1110_B	Reserved

3.5.3 System Registers

Figure 3-11 shows these system registers which include registers for:

The SBCU module includes the following TC21x/TC22x/TC23x standard system registers

- Register access protection

The following TC21x/TC22x/TC23x standard system registers are not included:

- Module Clock Control
- Module Kernel Reset
- OCDS Control and Status Register

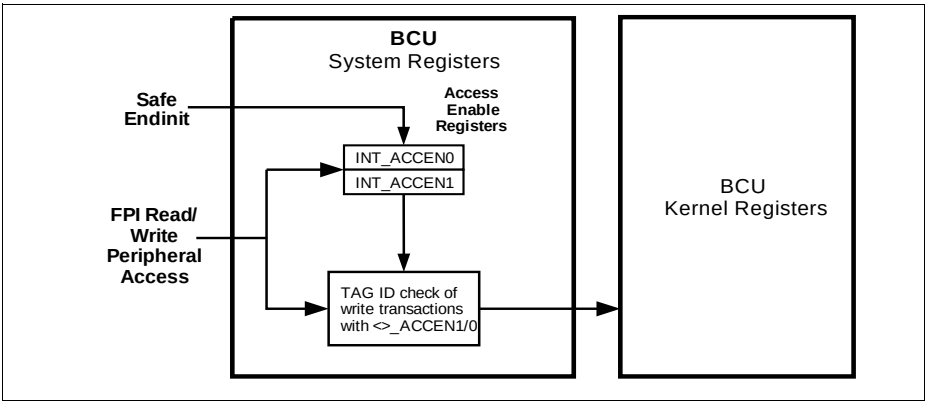


Figure 3-11 TC23 / TC22x OCB System System Registers

3.5.3.1 Register Access Protection (ACCEN1/0)

The TC23 / TC22x OCB System module provides a master TAG ID based write access protection as part of the AURIX safety concept. Each on chip resource with direct or indirect bus master capability has a unique master TAG ID that can be use to identify the master of an on chip bus transaction (see also chapter On Chip Bus Systems).

The SRC register is write protected via an On Chip Bus Master TAG-ID protection (see **Chapter 3.5.3.1**). This protection is controlled via the Interrupt Router control registers ACCEN10 and ACCEN00.

TAG ID based protection means that the support of write transactions to the TC23 / TC22x OCB System control registers can be enabled / disabled for each master TAG ID individually. For a disabled master TAG ID, write access will be disconnected with error acknowledge, read access will be processed (see also **Figure 3-11**).

The register access protection is controlled via the registers INT_ACCEN1 and INT_ACCEN0 where each bit is related to one encoding of the 6 bit On Chip Master TAG ID.

The INT_ACCEN1/0 registers are controlling the write access to all TC23 / TC22x OCB System control and system registers with the exception of the INT_ACCEN1/0 registers them self. INT_ACCEN1/0 are Safety Endinit protected.

After reset, all access enable bits and access control bits are enabled, access protection mechanism has to be configured and checked to bring the system in a safe state.

3.5.3.2 Kernel Reset Registers (KRST1/0, KRSTCLR)

The TC23 / TC22x OCB System module does not include the kernel reset registers (KRST1, KRST0, KRSTCLR).

Note: The TC23 / TC22x OCB System module does not support a module kernel reset.

3.5.3.3 Clock Control Register (CLC)

The TC23 / TC22x OCB System module does not include the module clock control (CLC).

Note: The TC23 / TC22x OCB System module does not support the Clock Control register functionality which means that the TC23 / TC22x OCB System module clock can not be disabled by the CLC register.

3.5.3.4 OCDS Control and Status Register (OCS)

The TC23 / TC22x OCB System module does not include OCDS Control and Status (OCS) register.

Note: The TC23 / TC22x OCB System module does not support the OCS register functionality.

3.5.4 BCU Debug Support

For debugging purposes, the BCU has the capability for breakpoint generation support. This OCDS debug capability is controlled by the Cerberus module and must be enabled by it (indicated by bit SBCU_DBCNTL.EO).

When BCU debug support has been enabled (EO = 1), any breakpoint request generated by the BCU to the Cerberus disarms the BCU breakpoint logic for further breakpoint requests. In order to rearm the BCU breakpoint logic again for the next breakpoint request generation, bit SBCU_DBCNTL.RA must be set. The status of the BCU breakpoint logic (armed or disarmed) is indicated by bit SBCU_DBCNTL.OA.

There are three types of trigger events:

- Address triggers
- Signal triggers
- Grant triggers

3.5.4.1 Address Triggers

The address debug trigger event conditions are defined by the contents of the SBCU_DBADR1, SBCU_DBADR2, and SBCU_DBCNTL registers. A wide range of possibilities arise for the creation of debug trigger events based on addresses. The following debug trigger events can be selected:

- Match on one signal address
- Match on one of two signal addresses
- Match on one address area
- Mismatch on one address area

Each pair of DBADRx registers and DBCNTL.ONAx bits determine one possible debug trigger event. The combination of these two possible debug trigger events defined by DBCNTL.CONCOM1 determine the address debug trigger event condition.

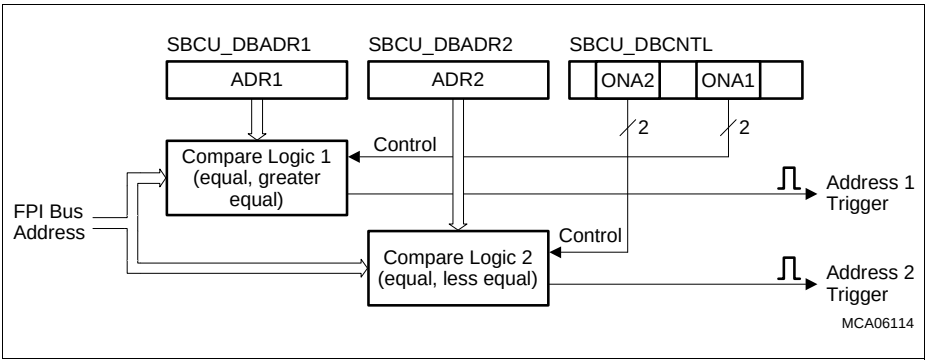


Figure 3-12 Address Trigger Generation

3.5.4.2 Signal Status Triggers

The signal status debug trigger event conditions are defined by the contents of the SBCU_DBBOS and SBCU_DBCNTL registers. Depending on the selected configuration a wide range of possibilities arise for the creation of a debug trigger event based on FPI Bus status signals. Possible combinations are:

- Match on a single signal status
- Match on a multiple signal status

With the multiple signal match conditions, all single signal match conditions are combined with a logical **AND** to the signal status debug trigger event signal. The selection whether or not a single match condition is selected can be enabled/disabled selectively for each condition via the SBCU_DBCNTL.ONBOSx bits.

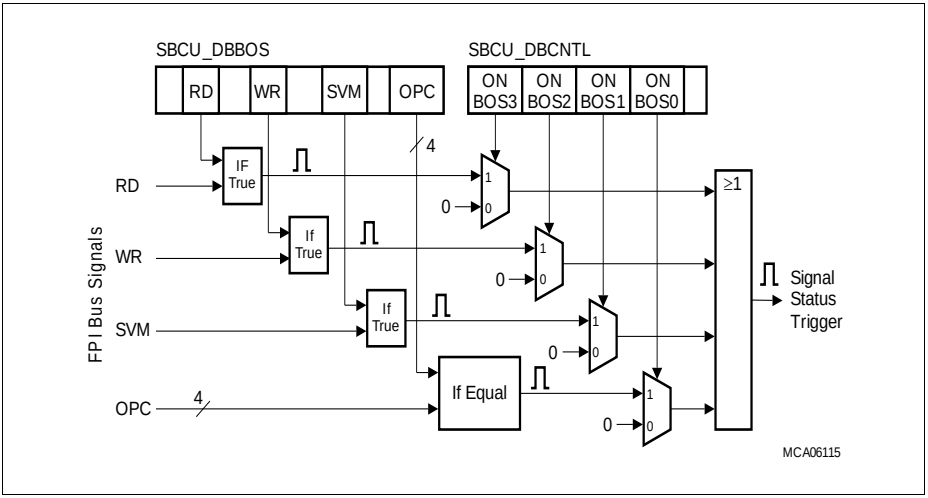


Figure 3-13 Signal Status Trigger Generation

3.5.4.3 Grant Triggers

The signal status debug trigger event conditions are defined via the registers SBCU_DBGRNT and SBCU_DBCNTL. Depending on the configuration of these registers, any combination of FPI Bus master trigger events can be configured. Only the enabled masters in the SBCU_DBGRNT register are of interest for the grant debug trigger event condition. The grant debug trigger event condition can be enabled/disabled via bit SBCU_DBCNTL.ONG (see **Figure 3-14**).

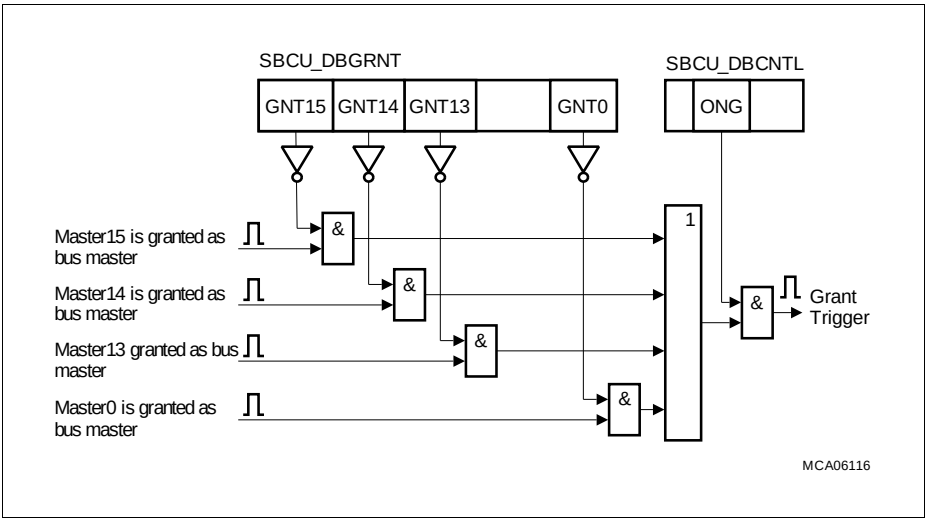


Figure 3-14 Grant Trigger Generation

3.5.4.4 Combination of Triggers

The combination of the four debug trigger signals to the single BCU breakpoint trigger event is defined via the bits CONCOM[2:0] of register SBCU_DBCNTL (see **Figure 3-15**). The two address triggers are combined to one address trigger that is further combined with signal status and grant trigger signals. A logical AND or OR combination can be selected for the BCU breakpoint trigger generation.

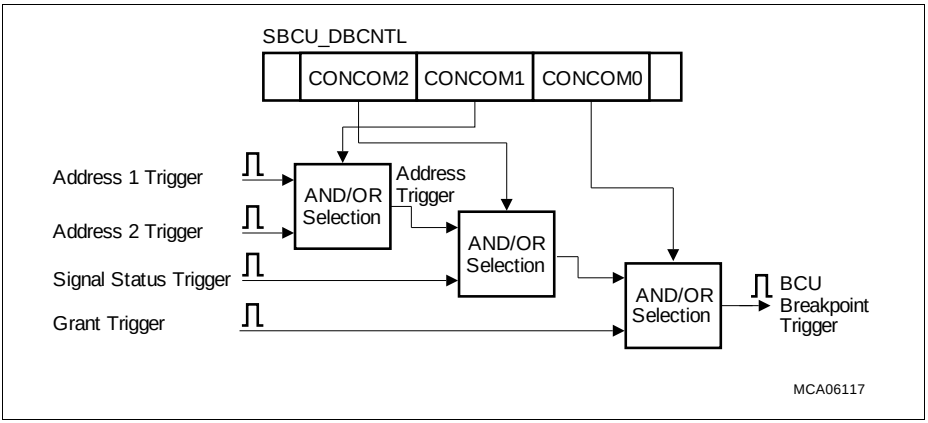


Figure 3-15 BCU Breakpoint Trigger Combination Logic

3.5.4.5 BCU Breakpoint Generation Examples

This section gives three examples of how BCU debug trigger events are programmed.

OCDS Debug Example 1

- Task: Generation of a BCU debug trigger event on any SPB write access to address 00002004_H or $000020A0_H$ by an SPB master.

For this task, the following programming settings for the BCU breakpoint logic must be executed:

1. Writing SBCU_DBADR1 = 0000 2004_H
2. Writing SBCU_DBADR2 = 0000 $20A0_H$
3. Writing SBCU_DBCNTL = $C1115010_H$:
   a) ONBOS[3:0] = 1100_B means that no signal status trigger is generated (disabled) for a read signal match AND write signal match condition according to the settings of bits RD and WR in register SBCU_DBBOS. Debug trigger event generation for Supervisor Mode signal match and op-code signal match condition is disabled.
   b) ONA2 = 01_B means that the equal match condition for debug address 2 register is selected.

c) ONA1 = 01_B means that the equal match condition for debug address 1 register is selected.
d) ONG = 1 means that the grant debug trigger is enabled.
e) CONCOM[2:0] = 101_B means that the address trigger is created by address trigger 1 OR address trigger 2 (CONCOM1 = 0), and that the grant trigger is ANDed with the address trigger (CONCOM0 = 1), and that the signal status trigger is ANDed with the address trigger (CONCOM2 = 1).
f) RA = 1 means that the BCU breakpoint logic is rearmed.

4. Writing SBCU_DBGRNT = $FFFFFFD7_H$:
means that the grant trigger for the SPB master is enabled.
5. Writing SBCU_DBBOS = 00001000_H:
means that the signal status trigger is generated on a write transfer and not on a read transfer.

OCDS Debug Example 2

- Task: generation of a BCU debug trigger event on any half-word access in User Mode to address area $01FFFFFF_H$ to $02FFFFFF_H$ by any master.

For this task, the following programming settings for the BCU breakpoint logic must be executed:

1. Writing SBCU_DBADR1 = $01FFFFFF_H$
2. Writing SBCU_DBADR2 = $02FFFFFF_H$
3. Writing SBCU_DBCNTL = 32206010_H:
a) ONBOS[3:0] = 0011_B means that the signal status trigger is disabled for a read or for write signal status match but enabled for Supervisor Mode match AND op-code match conditions according to the settings of bit SVM and bit field OPC in register SBCU_DBBOS.
b) ONA2 = 10_B means that the address 2 trigger is generated if the FPI Bus address is less or equal to SBCU_DBADR2.
c) ONA1 = 10_B means that the address 1 trigger is generated if the FPI Bus address is greater or equal to SBCU_DBADR1.
d) ONG = 0 means that the grant debug trigger is disabled.
e) CONCOM[2:0] = 110_B means that the address trigger is created by address trigger 1 AND address trigger 2 (CONCOM1 = 1), and that the grant trigger is OR-ed with the address trigger (CONCOM0 = 0), and that the signal status trigger is AND-ed with the address trigger (CONCOM2 = 1).
f) RA = 1 means that the BCU breakpoint logic is rearmed.
4. Writing SBCU_DBGRNT = $FFFFFFFF_H$:
means that no grant trigger for SPB masters is selected (“don’t care” because also disabled by ONG = 0).
5. Writing SBCU_DBBOS = 00000001_H:
means that the signal status trigger is generated for read (RD = 0) and write (WR = 0) half-word transfers (OPC = 0001_B) in User Mode (SVM = 0).

3.5.5 System Bus Control Unit Registers

Figure 3-16 and **Table 3-13** are showing the address maps with all registers of the System Bus Control Unit (SBCU) module.

List of used Reset Class abbreviations:

- Reset Class 1 -> Debug Reset (see description in the chapter SCU / Reset Types)
- Reset Class 3 -> Power On Reset, Application Reset, System Reset (see description in the chapter SCU / Reset Types)

SBCU Control Registers Overview

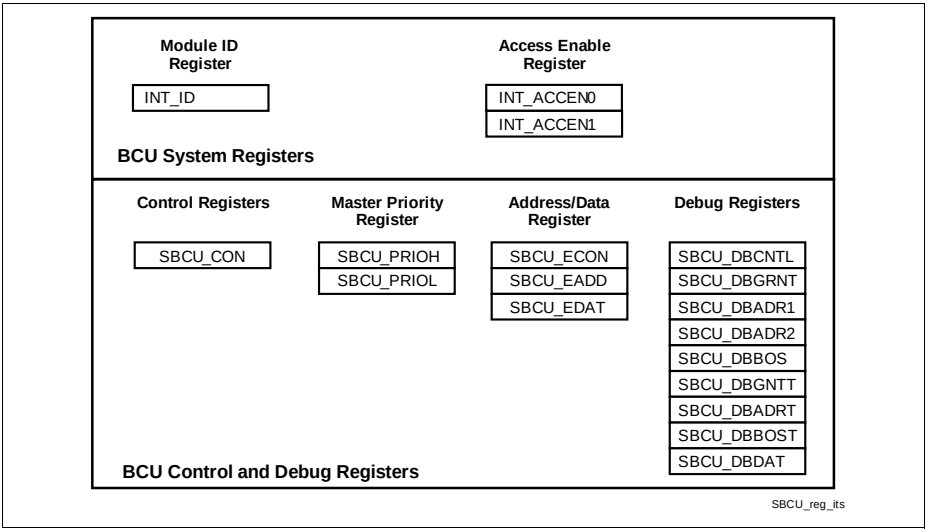


Figure 3-16 SBCU Registers

Table 3-12 Registers Address Space - SBCU Address Space

Module	Base Address	End Address	Note
SBCU	$F003\ 0000_H$	$F003\ 00FF_H$	

Table 3-13 Registers Overview - SBCU Control Registers

Short Name	Description	Offset Addr.[1)]	Access Mode		Reset Class	Description See
			Read	Write		
–	Reserved	000_H - 004_H	BE	BE	-	-
SBCU_ID	SBCU Module Identification Register	008_H	U, SV	BE	-	**Page 3-85**
–	Reserved	$00C_H$	BE	BE	-	-
SBCU_ CON	SBCU Control Register	010_H	U, SV	SV, P	3	**Page 3-88**
SBCU_ PRIOH	Arbiter Priority Register High	014_H	U, SV	SV, E, P	3	**Page 3-89**
SBCU_ PRIOL	Arbiter Priority Register Low	018_H	U, SV	SV, E, P	3	**Page 3-90**
–	Reserved	$01C_H$	BE	BE	3	-
SBCU_ ECON	SBCU Error Control Capture Register	020_H	U, SV	SV, P	3	**Page 3-92**
SBCU_ EADD	SBCU Error Address Capture Register	024_H	U, SV	SV, P	3	**Page 3-94**
SBCU_ EDAT	SBCU Error Data Capture Register	028_H	U, SV	SV, P	3	**Page 3-94**
–	Reserved	$02C_H$	BE	BE	-	-
SBCU_ DBCNTL	SBCU Debug Control Register	030_H	U, SV	SV, P	1	**Page 3-96**
SBCU_ DBGRNT	SBCU Debug Grant Mask Register	034_H	U, SV	SV, P	1	**Page 3-99**
SBCU_ DBADR1	SBCU Debug Address Register 1	038_H	U, SV	SV, P	1	**Page 3-101**
SBCU_ DBADR2	SBCU Debug Address Register 2	$03C_H$	U, SV	SV, P	1	**Page 3-101**
SBCU_ DBBOS	SBCU Debug Bus Operation Signals Register	040_H	U, SV	SV, P	1	**Page 3-102**
SBCU_ DBGNTT	SBCU Debug Trapped Master Register	044_H	U, SV	BE	1	**Page 3-103**

Table 3-13 Registers Overview - SBCU Control Registers

Short Name	Description	Offset Addr.[1]	Access Mode		Reset Class	Description See
			Read	Write		
SBCU_ DBADRT	SBCU Debug Trapped Address Register	048_H	U, SV	BE	1	Page 3-105
SBCU_ DBBOST	SBCU Debug Trapped Bus Operation Signals Register	$04C_H$	U, SV	BE	1	Page 3-106
SBCU_ DBDAT	SBCU Debug Data Status Register	050_H	U, SV	BE	1	Page 3-109
–	Reserved	054_H - $0F4_H$	BE	BE	-	-
SBCU_ ACCEN1	Access Enable Register 1 (Access Mode, Write: P1)	$0F8_H$	U, SV	SV, SE	3	Page 3-86
SBCU_ ACCEN0	Access Enable Register 0 (Access Mode, Write: P1)	$0FC_H$	U, SV	SV, SE	3	Page 3-87

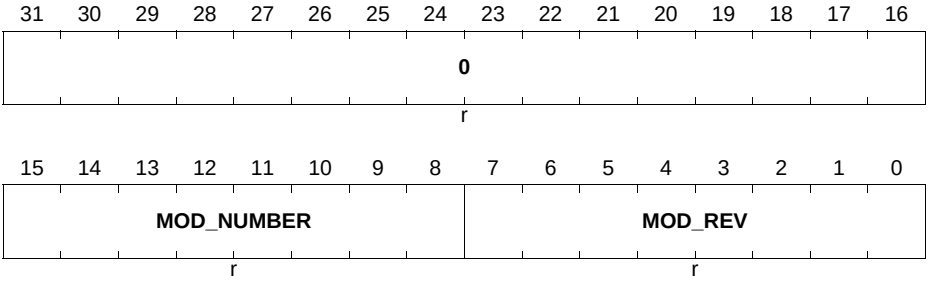
1) The absolute register address is calculated as follows:
Module Base Address (Table 3-12) + Offset Address (shown in this column)

3.5.5.1 SBCU System Registers

Module Identification Register (ID)

The identification register allows the programmer version-tracking of the module. The table below shows the identification register which is implemented in the SBCU module.

SBCU_ID
Module Identification Register **(008_H)** **Reset Value: 0000 6AXX$_H$**



Field	Bits	Type	Description
MOD_REV	[7:0]	r	**Module Revision Number** This bit field defines the module revision number. The value of a module revision starts with 01_H (first revision).
MOD_NUMBER	[15:8]	r	**Module Number Value** This bit field defines a module identification number. The value for the LBCU module is 006AH.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

Access Enable Register 0 (ACCEN0)

The Access Enable Register 0 controls write access for transactions with the on chip bus master TAG ID 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The registers ACCEN00 / ACCEN01 are providing one enable bit for each 6-bit On Chip Bus Master TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000_B, EN1 -> TAG ID 000001_B , ... ,EN31 -> TAG ID 011111_B.

SBCU_ACCEN0
Access Enable Register 0 ($0FC_H$) Reset Value: $FFFF\ FFFF_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	EN30	EN29	EN28	EN27	EN26	EN25	EN24	EN23	EN22	EN21	EN20	EN19	EN18	EN17	EN16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

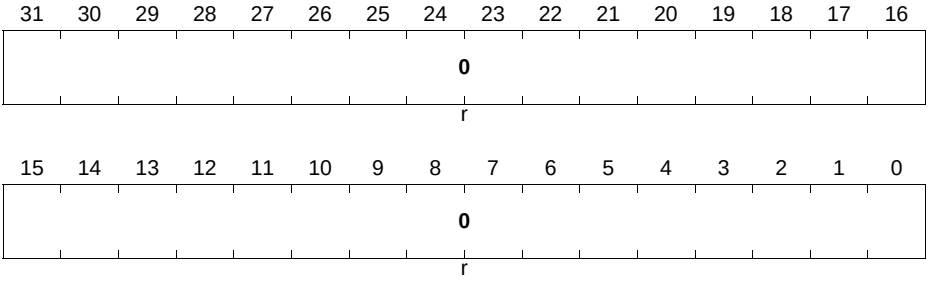
Field	Bits	Type	Description
ENn (n = 0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Access Enable Register 1 (ACCEN1)

The Access Enable Register 1 controls write access for transactions with the on chip bus master TAG ID 100000_B to 111111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). ACCEN11/01 are not implemented with register bits as the related On Chip Bus Master TAG IDs are not used in the AURIX devices.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001B, ... ,EN31 -> TAG ID 111111B.

SBCU_ACCEN1
Access Enable Register 1 **($0F8_H$)** **Reset Value: $0000\ 0000_H$**

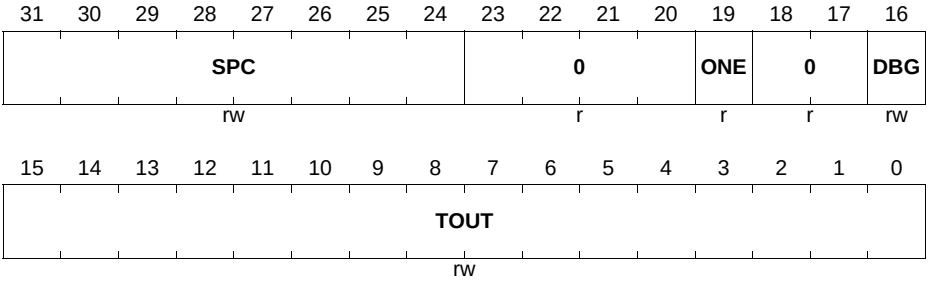


Field	Bits	Type	Description
0	[31:0]	r	**Reserved** Read as 0; should be written with 0.

3.5.5.2 SBCU Control Registers Descriptions

The SBCU Control Register controls the overall operation of the SBCU, including setting the starvation sample period, the bus time-out period, enabling starvation-protection mode, and error handling.

SBCU_CON
SBCU Control Register **(010_H)** **Reset Value: FF09 FFFF$_H$**

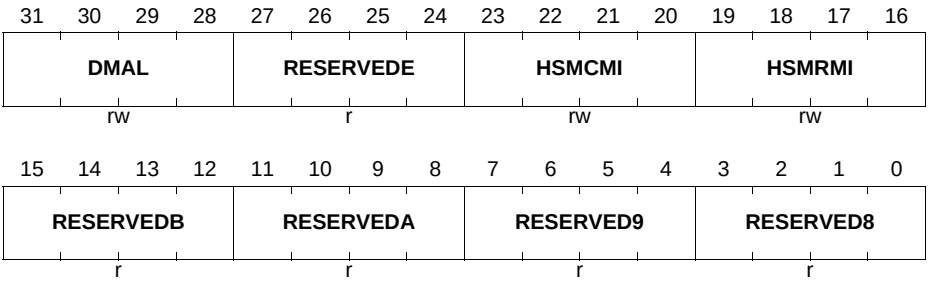


Field	Bits	Type	Description
TOUT	[15:0]	rw	**SBCU Bus Time-Out Value** The bit field determines the number of System Peripheral Bus time-out cycles. Default after reset is FFFF$_H$ (= 65536 bus cycles). Please Note: TOUT value must be >= 5.
DBG	16	rw	**SBCU Debug Trace Enable** The bit enables/disables the error capture mechanism for the registers BCU_ECON, BCU_EADD, BCU_EDAT 0_B SBCU debug trace disabled 1_B SBCU debug trace enabled (default after reset) *Note: The bit does not affect the SMU alarm or the BCU interrupt that are send in case of an error condition.*
SPC	[31:24]	rw	**Starvation Period Control** Determines the sample period for the starvation counter. Must be larger than the number of masters. The reset value is FF$_H$.
ONE	19	r	**Reserved** Read as 1; should be written with 0.

Field	Bits	Type	Description
0	[23:20], [18:17]	r	**Reserved** Read as 0; should be written with 0.

SBCU_PRIOH

Arbiter Priority Register High **(014_H)** **Reset Value: FEDC BA98$_H$**



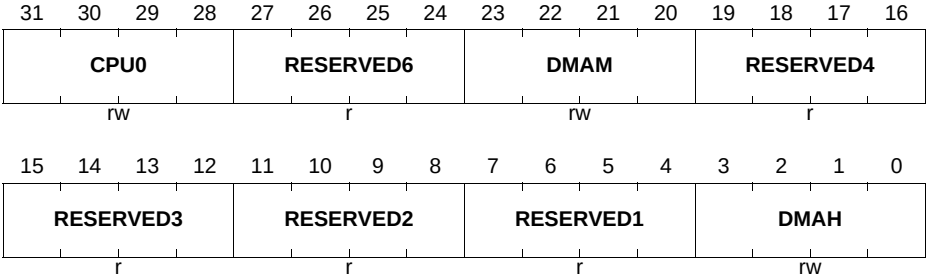
Field	Bits	Type	Description
DMAL	[31:28]	rw	**Master 15 Priority**[1)] This bit field contains the master priority for master connected to BCU request input 15 A lower number has a higher priority in the arbitration round than a higher one.
HSMRMI	[19:16]	rw	**Master 12 Priority** This bit field contains the master priority for master connected to BCU request input 12 A lower number has a higher priority in the arbitration round than a higher one.
HSMCMI	[23:20]	rw	**Master 13 Priority** This bit field contains the master priority for master connected to BCU request input 13 A lower number has a higher priority in the arbitration round than a higher one.

Field	Bits	Type	Description
RESERVEDE, **RESERVEDB,** **RESERVEDA,** **RESERVED9,** **RESERVED8**	[27:24], [15:12], [11:8], [7:4], [3:0]	r	**Reserved** Read as 1; should be written with 0.

1) Including DMA transactions from DMA channels with low priority, MLI and from Cerberus with low priority.

SBCU_PRIOL

Arbiter Priority Register Low **(018_H)** **Reset Value: 7654 3210$_H$**



Field	Bits	Type	Description
DMAH	[3:0]	rw	**Master 0 Priority**[1)] This bit field contains the master priority for master connected to BCU request input 0 A lower number has a higher priority in the arbitration round than a higher one.
DMAM	[23:20]	rw	**Master 5 Priority**[2)] This bit field contains the master priority for master connected to BCU request input 5 A lower number has a higher priority in the arbitration round than a higher one.
CPU0	[31:28]	rw	**Master 7 Priority** This bit field contains the master priority for master connected to BCU request input 8 A lower number has a higher priority in the arbitration round than a higher one.

Field	Bits	Type	Description
RESERVED6, RESERVED4, RESERVED3, RESERVED2, RESERVED1	[27:24], [19:16], [15:12], [11:8], [7:4]	r	**Reserved** Read as 1; should be written with 0.

1) Including DMA transactions from DMA channels with high priority and from Cerberus with high priority.

2) Including DMA transactions from DMA channels with medium priority.

Note: Reset values for bits/bit fields coupled to masters or slaves that are not configured or enabled are zero

3.5.5.3 SBCU Error Registers Descriptions

The capture of bus error conditions is enabled by setting SBCU_CON.DBG to 1. In case of a bus error, information about the condition will then be stored in the SBCU error capture registers. The SBCU error capture registers can then be examined by software to determine the cause of the FPI Bus error.

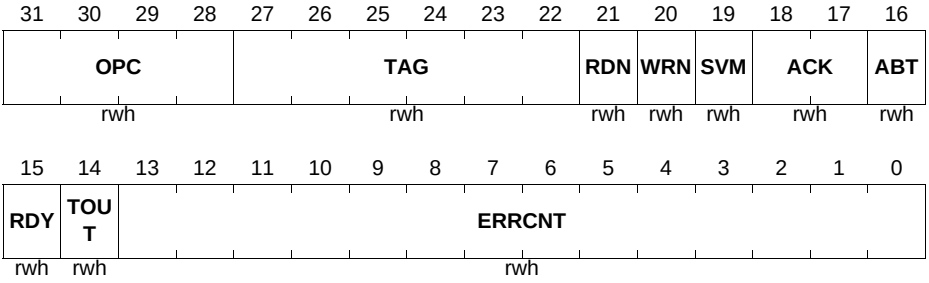
If enabled and an FPI Bus error occurs, the SBCU_ECON register holds the captured FPI Bus control information and an error count of the number of bus errors. The SBCU_EADD register stores the captured FPI Bus address. The SBCU_EDAT register stores the captured FPI Bus data.

If the capture of FPI Bus error conditions is disabled (SBCU_CON.DBG = 0), the SBCU error capture registers remain untouched.

Note: The SBCU error capture registers store only the parameters of the first error. In case of multiple bus errors, an error counter SBCU_ECON.ERRCNT shows the number of bus errors since the first error occurred. An application reset clears this bit field to zero, but the counter can be set to any value through software. This counter is prevented from overflowing, so a value of 2^{16} - 1 indicates that at least this many errors have occurred, but there may have been more. After SBCU_ECON has been read, the SBCU_ECON, SBCU_EADD and SBCU_EDAT registers are re-enabled to trace FPI Bus error conditions.

SBCU_ECON

SBCU Error Control Capture Register (020_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
ERRCNT	[13:0]	rwh	**FPI Bus Error Counter** ERRCNT is incremented on every occurrence of an FPI Bus error. ERRCNT is reset to 0 after the SBCU_ECON register is read.[1)]
TOUT	14	rwh	**State of FPI Bus Time-Out Signal** This bit indicates the state of the time-out signal at an FBI Bus error. 0_B No time-out occurred 1_B Time-out has occurred
RDY	15	rwh	**State of FPI Bus Ready Signal** This bit indicates the state of the ready signal at an FBI Bus error. 0_B Wait state(s) have been inserted. Ready signal was active 1_B Ready signal was inactive
ABT	16	rwh	**State of FPI Bus Abort Signal** This bit indicates the state of the abort signal at an FBI Bus error. 0_B Master has aborted an FPI Bus transfer. Abort signal was active 1_B Abort signal was inactive
ACK	[18:17]	rwh	**State of FPI Bus Acknowledge Signals** This bit field indicates the acknowledge code that has been output by the selected slave at an FPI Bus error. Coding see **Table 3-10**.

Field	Bits	Type	Description
SVM	19	rwh	**State of FPI Bus Supervisor Mode Signal** This bit indicates whether the FPI Bus error occurred in Supervisor Mode or in User Mode. 0_B Transfer was initiated in User Mode 1_B Transfer was initiated in Supervisor Mode
WRN	20	rwh	**State of FPI Bus Write Signal** This bit indicates whether the FPI Bus error occurred at a write cycle (see**Table 3-14**).
RDN	21	rwh	**State of FPI Bus Read Signal** This bit indicates whether the FPI Bus error occurred at a read cycle (see**Table 3-14**).
TAG	[27:22]	rwh	**FPI Bus Master Tag Number Signals** This bit field indicates the FPI Bus master TAG number (definitions see **Table 3-15**).
OPC	[31:28]	rwh	**FPI Bus Operation Code Signals** The FPI Bus operation codes are defined in **Table 3-11**.

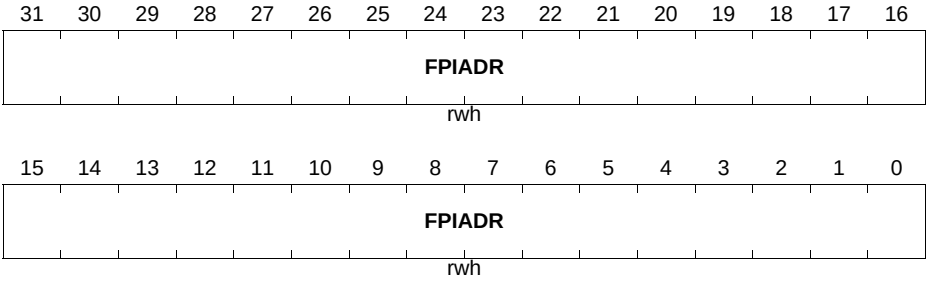
1) In the TC21x/TC22x/TC23x, aborted accesses to a 0 wait state SPB slave may also increment ERRCNT when the slave generates an error acknowledge.

Table 3-14 FPI Bus Read/Write Error Indication

$\overline{RD}$	$\overline{WR}$	FPI Bus Cycle
0	0	FPI Bus error occurred at the read transfer of a read-modify-write transfer.
0	1	FPI Bus error occurred at a read cycle of a single transfer.
1	0	FPI Bus error occurred at a write cycle of a single transfer or at the write cycle of a read-modify-write transfer.
1	1	Does not occur.

SBCU_EADD

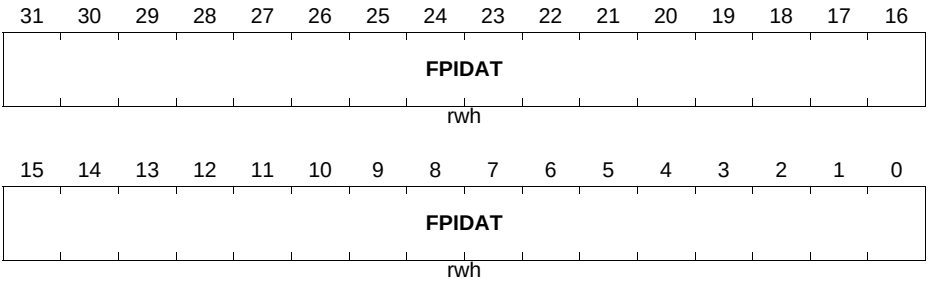
SBCU Error Address Capture Register (024_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
FPIADR	[31:0]	rwh	**Captured FPI Bus Address** This bit field holds the 32-bit FPI Bus address that has been captured at an FPI Bus error. Note that if multiple bus errors occurred, only the address of the first bus error is captured.

SBCU_EDAT

SBCU Error Data Capture Register (028_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
FPIDAT	[31:0]	rwh	**Captured FPI Bus Data** This bit field holds the 32-bit FPI Bus data that has been captured at an FPI Bus error. Note that if multiple bus errors occurred, only the data of the first bus error is captured.

3.5.5.4 SBCU OCDS Registers Descriptions

SBCU_DBCNTL
SBCU Debug Control Register **(030_H)** **Reset Value: 0000 7003$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ONBOS3	ONBOS2	ONBOS1	ONBOS0	0		ONA2		0		ONA1		0			ONG
rw	rw	rw	rw	r		rw		r		rw		r			rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CONCOM2	CONCOM1	CONCOM0	0							RA	0		OA	EO
r	rw	rw	rw	r							w	r		r	r

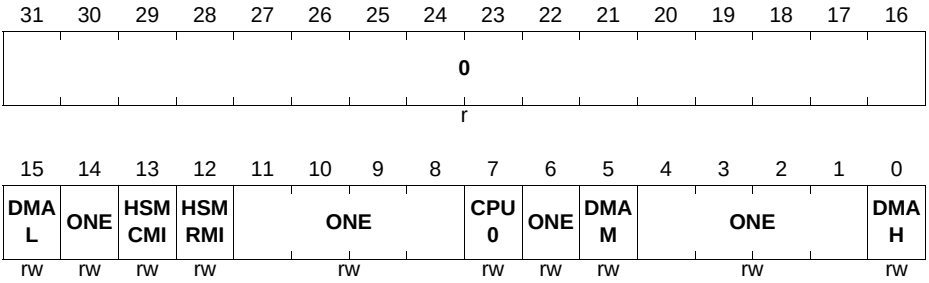
Field	Bits	Type	Description
EO	0	r	**Status of SBCU Debug Support Enable** This bit is controlled by the Cerberus and enables the SBCU debug support. 0_B SBCU debug support is disabled 1_B SBCU debug support is enabled (default after reset)
OA	1	r	**Status of SBCU Breakpoint Logic** 0_B The SBCU breakpoint logic is disarmed. Any further breakpoint activation is discarded 1_B The SBCU breakpoint logic is armed The OA bit is set by writing a 1 to bit RA. When OA is set, registers SBCU_DBGNTT, SBCU_DBADRT, and SBCU_DBDAT are reset. Also SBCU_DBBOST is reset with the exception of the bit field FPIRST.
RA	4	w	**Rearm SBCU Breakpoint Logic** Writing a 1 to this bit rearms SBCU breakpoint logic and sets bit OA = 1. RA is always reads as 0.

Field	Bits	Type	Description
CONCOM0	12	rw	**Grant and Address Trigger Relation** 0_B The grant phase trigger condition and the address trigger condition (see CONCOM1) are combined with a logical OR for further control 1_B The grant phase trigger condition and the address trigger condition (see CONCOM1) are combined with a logical AND for further control (see **Figure 3-15**)
CONCOM1	13	rw	**Address 1 and Address 2 Trigger Relation** 0_B Address 1 trigger condition and address 2 trigger condition are combined with a logical OR to the address trigger condition for further control 1_B Address 1 trigger condition and address 2 trigger condition are combined with a logical AND to the address trigger condition for further control (see **Figure 3-15**)
CONCOM2	14	rw	**Address and Signal Trigger Relation** 0_B Address trigger condition (see CONCOM1) and signal status trigger conditions are combined with a logical OR for further control 1_B Address phase trigger condition (see CONCOM1) and the signal status trigger conditions are combined with a logical AND for further control (see **Figure 3-15**)
ONG	16	rw	**Grant Trigger Enable** 0_B No grant debug event trigger is generated 1_B The grant debug event trigger is enabled and generated according the settings of register SBCU_DBGRNT (see **Figure 3-15**)
ONA1	[21:20]	rw	**Address 1 Trigger Control** 00_B No address 1 trigger is generated 01_B An address 1 trigger event is generated if the FPI Bus address is equal to SBCU_DBADR1 10_B An address 1 trigger event is generated if FPI Bus address is greater or equal to SBCU_DBADR1 11_B same as 00_B (See also **Figure 3-12**).

Field	Bits	Type	Description
ONA2	[25:24]	rw	**Address 2 Trigger Control** 00_B No address 2 trigger is generated 01_B An address 2 trigger event is generated if the FPI Bus address is equal to SBCU_DBADR2 10_B An address 2 trigger event is generated if FPI Bus address is less or equal to SBCU_DBADR2 11_B same as 00_B See also **Figure 3-12**.
ONBOS0	28	rw	**Opcode Signal Status Trigger Condition** 0_B A signal status trigger is generated for all FPI Bus op-codes except a “no operation” op-code 1_B A signal status trigger is generated if the FPI Bus op-code matches the op-code as defined in DBBOS.OPC (see **Figure 3-13**)
ONBOS1	29	rw	**Supervisor Mode Signal Trigger Condition** 0_B The signal status trigger generation for the FPI Bus Supervisor Mode signal is disabled 1_B A signal status trigger is generated if the FPI Bus Supervisor Mode signal state is equal to the value of DBBOS.SVM (see **Figure 3-13**)
ONBOS2	30	rw	**Write Signal Trigger Condition** 0_B The signal status trigger generation for the FPI Bus write signal is disabled 1_B A signal status trigger is generated if the FPI Bus write signal state is equal to the value of DBBOS.WR (see **Figure 3-13**)
ONBOS3	31	rw	**Read Signal Trigger Condition** 0_B The signal status trigger generation for the FPI Bus read signal is disabled 1_B A signal status trigger is generated if the FPI Bus read signal state is equal to the value of DBBOS.RD (see **Figure 3-13**)

Field	Bits	Type	Description
0	[3:2], [11:5], 15, [19:17], [23:22], [27:26]	r	**Reserved** Read as 0; should be written with 0.

SBCU_DBGRNT
SBCU Debug Grant Mask Register (034_H) Reset Value: $0000\ FFFF_H$



Field	Bits	Type	Description
DMAH	0	rw	**Cerberus Grant Trigger Enable, High Priority[1)]** 0_B FPI Bus transactions with high-priority DMA as bus master are enabled for grant trigger event generation 1_B FPI Bus transactions with high-priority DMA as bus master are disabled for grant trigger event generation
DMAM	5	rw	**DMA Grant Trigger Enable, Medium Priority[2)]** 0_B FPI Bus transactions with medium-priority DMA channels as bus master are enabled for grant trigger event generation 1_B FPI Bus transactions with medium-priority DMA channels as bus master are disabled for grant trigger event generation

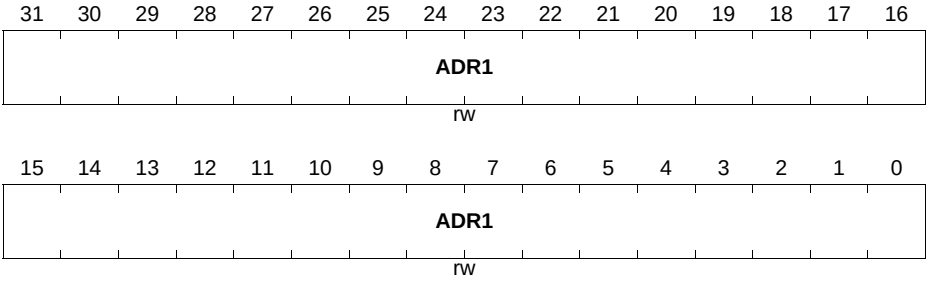
Field	Bits	Type	Description
CPU0	7	rw	**CPU0 Grant Trigger Enable** 0_B FPI Bus transactions with CPU0 as bus master are enabled for grant trigger event generation 1_B FPI Bus transactions with CPU as bus master are disabled for grant trigger event generation
HSMRMI	12	rw	**HSM Register Master Interface Grant Trigger Enable** 0_B FPI Bus transactions requested by the HSM bus master are enabled for grant trigger event generation 1_B FPI Bus transactions requested by the HSM bus master are disabled for grant trigger event generation
HSMCMI	13	rw	**HSM Cache Master Interface Grant Trigger Enable** 0_B FPI Bus transactions requested by the HSM bus master are enabled for grant trigger event generation 1_B FPI Bus transactions requested by the HSM bus master are disabled for grant trigger event generation
DMAL	15	rw	**DMA Grant Trigger Enable, Low Priority[3)]** 0_B FPI Bus transactions with low-priority DMA channels as bus master are enabled for grant trigger event generation 1_B FPI Bus transactions with low-priority DMA channels as bus master are disabled for grant trigger event generation
ONE, ONE, ONE, ONE	14, [11:8], 6, [4:1]	rw	**Reserved** Read as 1 after reset; reading these bits will return the value last written.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

1) Including DMA transactions from DMA channels with high priority and from Cerberus with high priority.

2) Including DMA transactions from DMA channels with medium priority.

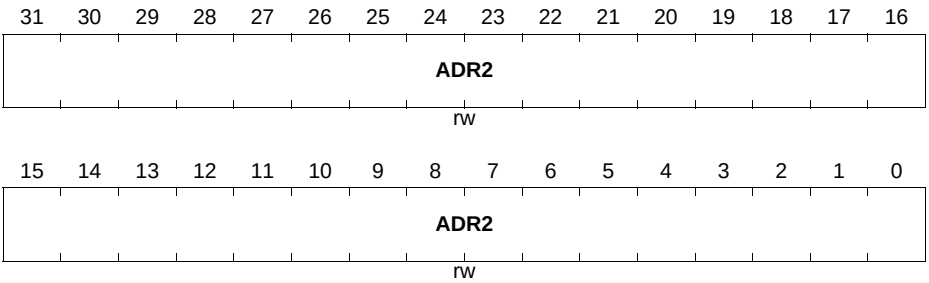
3) Including DMA transactions from DMA channels with low priority and from Cerberus with low priority.

SBCU_DBADR1
SBCU Debug Address 1 Register (038_H) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
ADR1	[31:0]	rw	**Debug Trigger Address 1** This register contains the address for the address 1 trigger event generation.

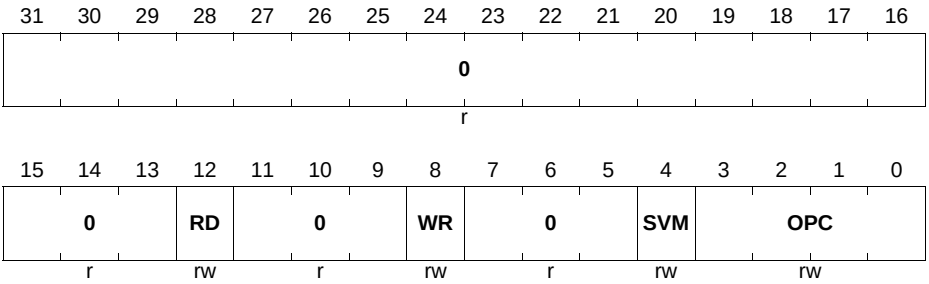
SBCU_DBADR2
SBCU Debug Address 2 Register ($03C_H$) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
ADR2	[31:0]	rw	**Debug Trigger Address 2** This register contains the address for the address 2 trigger event generation.

SBCU_DBBOS
SBCU Debug Bus Operation Signals Register

(040_H) Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
OPC	[3:0]	rw	**Opcode for Signal Status Debug Trigger** This bit field determines the type (opcode) of an FPI Bus transaction for which a signal status debug trigger event is generated (if enabled by DBCNTL.ONBOS0 = 1). 0000_B Trigger on single byte transfer selected 0001_B Trigger on single half-word transfer selected 0010_B Trigger on single word transfer selected 0100_B Trigger on 2-word block transfer selected 0101_B Trigger on 4-word block transfer selected 0110_B Trigger on 8-word block transfer selected 1111_B Trigger on no operation selected Other bit combinations are reserved.
SVM	4	rw	**SVM Signal for Status Debug Trigger** This bit determines the mode of an FPI Bus transaction for which a signal status debug trigger event is generated (if enabled by DBCNTL.ONBOS1 = 1). 0_B Trigger on User Mode selected 1_B Trigger on Supervisor Mode selected

Field	Bits	Type	Description
WR	8	rw	**Write Signal for Status Debug Trigger** This bit determines the state of the $\overline{WR}$ signal of an FPI Bus transaction for which a signal status debug trigger event is generated (if enabled by DBCNTL.ONBOS2 = 1). 0_B Trigger on a single write transfer or write cycle of an atomic transfer selected 1_B No operation or read transaction selected
RD	12	rw	**Write Signal for Status Debug Trigger** This bit determines the state of the $\overline{RD}$ signal of an FPI Bus transaction for which a signal status debug trigger event is generated (if enabled by DBCNTL.ONBOS3 = 1). 0_B Trigger on a single read transfer or read cycle of an atomic transfer selected 1_B No operation or write transfer selected
0	[7:5], [11:9], [31:13]	r	**Reserved** Read as 0; should be written with 0.

SBCU_DBGNTT

SBCU Debug Trapped Master Register

(044_H) **Reset Value: 0000 FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ONE								DMACHNR							
rw								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAL	ONE	HSMCMI	HSMRMI	ONE				CPU0	ONE	DMAM	ONE				DMAH
rw	rw	rw	rw	rw				rw	rw	rw	rw				rw

Field	Bits	Type	Description
DMAH	0	rw	**High-Priority DMA FPI Bus Master Status[1)]** This bit indicates whether the DMA with a medium priority request was FPI Bus master when the break trigger event occurred. 0_B The medium-priority DMA was the FPI bus master. 1_B The medium-priority DMA was not the FPI Bus master.
DMAM	5	rw	**Medium-Priority DMA FPI Bus Master Status[2)]** This bit indicates whether the DMA with a medium priority request was FPI Bus master when the break trigger event occurred. 0_B The medium-priority DMA was the FPI bus master. 1_B The medium-priority DMA was not the FPI Bus master.
CPU0	7	rw	**CPU0 FPI Bus Master Status** This bit indicates whether the CPU0 was FPI Bus master when the break trigger event occurred. 0_B The CPU0 was the FPI Bus master. 1_B The CPU0 was not the FPI Bus master.
HSMRMI	12	rw	**HSM Register FPI Bus Master Interface Status** This bit indicates whether the HSM was FPI Bus master when the break trigger event occurred. 0_B HSMRMI was the FPI bus master. 1_B HSMRMI was not the FPI Bus master.
HSMCMI	13	rw	**HSM Cache FPI Bus Master Interface Status** This bit indicates whether the HSM was FPI Bus master when the break trigger event occurred. 0_B HSMCMI was the FPI bus master. 1_B HSMCMI was not the FPI Bus master.

Field	Bits	Type	Description
DMAL	15	rw	**Low-Priority DMA FPI Bus Master Status**[3)] This bit indicates whether the DMA with a low-priority request was the FPI Bus master when the break trigger event occurred. 0_B The low-priority DMA was the FPI Bus master. 1_B The low-priority DMA was not the FPI Bus master.
DMACHNR	[23:16]	rw	**DMA-FPI Channel Grant Status** The encoding is directly dedicated to the dma channel number
ONE, **ONE,** **ONE,** **ONE,** **ONE**	[31:24], 14, [11:8], 6, [4:1]	rw	**Reserved** Read as 1; shall be written with 0.

1) Including DMA transactions from DMA channels with high priority and from Cerberus with high priority.

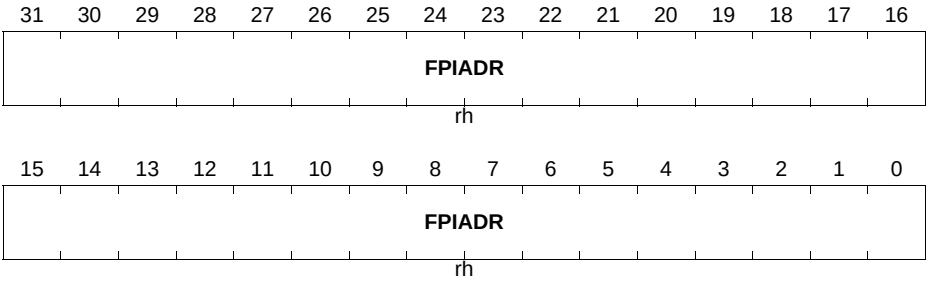
2) Including DMA transactions from DMA channels with medium priority.

3) Including DMA transactions from DMA channels with low priority, MLI and from Cerberus with low priority.

SBCU_DBADRT

SBCU Debug Trapped Address Register

(048_H) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
FPIADR	[31:0]	rh	**FPI Bus Address Status** This register contains the FPI Bus address that was captured when the OCDS break trigger event occurred.

SBCU_DBBOST
SBCU Debug Trapped Bus Operation Signals Register
(04C$_H$) **Reset Value: 0000 3180$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0										FPITAG					
rh										rh					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENDINIT	FPITOUT	FPIABORT	FPIRD	FPIOPS	FPIRST		FPIWR	FPIRDY	FPIACK		FPISVM	FPIOPC			
rh	rh	rh	rh	rh	rh		rh	rh	rh		rh	rh			

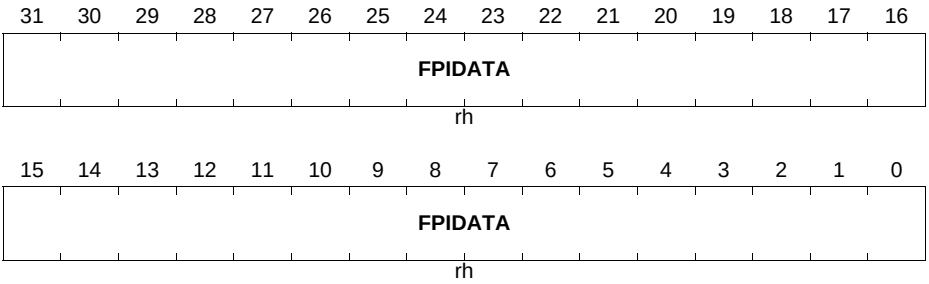
Field	Bits	Type	Description
FPIOPC	[3:0]	rh	**FPI Bus Opcode Status** This bit field indicates the type (opcode) of the FPI Bus transaction captured from the FPI Bus signal lines when the BCU break trigger event occurred. 0000_B Single byte transfer 0001_B Single half-word transfer 0010_B Single word transfer 0100_B 2-word block transfer 0101_B 4-word block transfer 0110_B 8-word block transfer 1111_B No operation Other bit combinations are reserved.

Field	Bits	Type	Description
FPISVM	4	rh	**FPI Bus Supervisor Mode Status** This bit indicates the state of the Supervisor Mode signal captured from the FPI Bus signal lines when the BCU break trigger event occurred. 0_B User mode 1_B Supervisor mode
FPIACK	[6:5]	rh	**FPI Bus Acknowledge Status** This bit field indicates the acknowledge signal status captured from the FPI Bus signal lines when the BCU break trigger event occurred. 00_B No special case 01_B Error 10_B Reserved 11_B Retry, slave did not respond
FPIRDY	7	rh	**FPI Bus Ready Status** This bit indicates the ready signal status captured from the FPI Bus signal lines when the BCU break trigger event occurred. 0_B Last cycle of transfer 1_B Not last cycle of transfer
FPIWR	8	rh	**FPI Bus Write Indication Status** This bit indicates the write signal status captured from the FPI Bus signal lines when the BCU break trigger event occurred. 0_B Single write transfer or write cycle of an atomic transfer 1_B No operation or read transfer
FPIRST	[10:9]	rh	**FPI Bus Reset Status** This bit field indicates the reset signal status captured from the FPI Bus signal lines when the BCU break trigger event occurred. 00_B Reset of all FPI Bus components 11_B No reset Others Reserved

Field	Bits	Type	Description
FPIOPS	11	rh	**FPI Bus OCDS Suspend Status** This bit indicates the OCDS suspend signal status captured from the FPI Bus signal lines when the BCU break trigger event occurred. 0_B No OCDS suspend request is pending 1_B An OCDS suspend request is pending
FPIRD	12	rh	**FPI Bus Read Indication Status** This bit indicates the read signal status captured from the FPI Bus signal lines when the BCU break trigger event occurred. 0_B Single read transfer or read cycle of an atomic transfer 1_B No operation or write transfer
FPIABORT	13	rh	**FPI Bus Abort Status** This bit indicates the abort signal status captured from the FPI Bus signal lines when the BCU break trigger event occurred. 0_B A transfer that has already started was aborted 1_B Normal operation
FPITOUT	14	rh	**FPI Bus Time-out Status** This bit indicates the time-out signal status captured from the FPI Bus signal lines when the BCU break trigger event occurred. 0_B Normal operation 1_B A time-out event was generated
ENDINIT	15	rh	**FPI Bus Endinit Status** This bit indicates the ENDINIT signal status captured from the FPI Bus signal lines when the BCU break trigger event occurred. 0_B Normal operation 1_B System was in ENDINIT state
FPITAG	[21:16]	rh	**FPI Bus Master TAG Status** This bit field indicates the master TAG captured from the FPI Bus signal lines when the BCU break trigger event occurred (see **Table 3-15**). The master TAG identifies the master of the transfer which generated BCU break trigger event.
0	[31:22]	rh	**Reserved** Read as 0; should be written with 0.

SBCU_DBDAT
SBCU Debug Data Status Register

(050_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
FPIDATA	[31:0]	rh	**FPI Bus Data Status** This register contains the FPI Bus data that was captured when the OCDS break trigger event occurred.

3.6 On Chip Bus Master TAG Assignments

Each master interface on the System Peripheral Bus and on the SRI Bus is assigned to a 6-bit identification number, the master TAG number (see **Table 3-15**). This makes it possible for software debug and MCDS purposes to distinguish which master has performed the current transaction (see **"XBAR_ERR0" on Page 3-51**[1)] and **"SBCU_DBADRT" on Page 3-105**).

Table 3-15 On Chip Bus Master TAG Assignments

TAG-Number	Module	Location	Description
000000_B	CPU0	SRI/SPB	DMI.NonSafe TAG ID
000001_B	CPU0	SRI/SPB	DMI.Safe TAG ID
000010_B	-	-	Reserved
000011_B	-	-	Reserved
000100_B	-	-	Reserved

1) Pls. note that the Transaction ID bit field in the register **"XBAR_ERR0" on Page 3-51** includes the TAG ID on the 6 MSB (TRID[5:0]).

Table 3-15 On Chip Bus Master TAG Assignments

TAG-Number	Module	Location	Description
000101_B	-	-	Reserved
000110_B	DMA	SRI/SPB	DMA Resource Partition 0
000111_B	DMA	SRI/SPB	DMA Resource Partition 1
001000_B	DMA	SRI/SPB	DMA Resource Partition 2
001001_B	DMA	SRI/SPB	DMA Resource Partition 3
001010_B	DMA	SRI/SPB	Cerberus
001011_B	-	-	Reserved
001100_B	Ethernet	SPB	TC23x ED: Ethernet TC23x: Reserved TC22x: Reserved
001101_B	HSM	SPB	TC23x: HSMCMI, HSMRMI[1)] TC22x: Reserved
001110_B	-	-	Reserved
001111_B	-	-	Reserved
010000_B	CPU0	SRI	PMI
010001_B	-	-	Reserved
011100_B	IOC32	BBB	Cerberus on Back Bone Bus (ED)
011101_B	EMEM	BBB	EMEM Master on Back Bone Bus (ED)
Others	-	-	Reserved

1) Both HSM FPI Master Interfaces (HSMCMI, HSMRMI) are using the sam TAG ID

3.7 On Chip Bus Access Times

The table describes the CPU access times in CPU clock cycles for the TC21x/TC22x/TC23x. The access times are described as maximum CPU stall cycles where e.g. an data access to the local DSPR results in zero stall cycles. Pls. note that the CPU does not always immediately stall after the start of a data read from another SPR due to instruction pipelining effects. This means that the average number will be below the here shown numbers[RF00105].

Note: All values are preliminary

Table 3-16 CPU access latency in CPU clock cycles for TC21x/TC22x/TC23x

CPU Access Mode	CPU clock cycles
Data read access to own DSPR	0
Data write access to own DSPR	0
Data read access to own or other PSPR	5
Data write access to own or other PSPR	0
Data read access to own or other DSPR	5
Data write access to own or other DSPR	0
Instruction fetch from own PSPR	0
Instruction fetch from other PSPR (critical word)	5
Instruction fetch from other PSPR (any remaining words)	0
Instruction fetch from other DSPR (critical word)	5
Instruction fetch from other DSPR (any remaining words)	0
Initial Pflash Access (critical word)	5 + configured PFlash Wait States[1)]
Initial Pflash Access (remaining words)	0
PMU PFlash Buffer Hit (critical word)	4
PMU PFlash Buffer Hit (remaining words)	0
Initial Dflash Access	5 + configured DFlash Wait States[2)]
TC1.6E/P Data read from System Peripheral Bus (SPB)	4 ($f_{CPU}=f_{SPB}$) 7 ($f_{CPU}=2*f_{SPB}$)
TC1.6E/P Data write to System Peripheral Bus (SPB)	0

1) FCON.WSPFLASH + FCON.WSECPF (see PMU chapter for the detailed description of these parameters).

2) FCON.WSDFLASH + FCON.WSECDF (see PMU chapter for the detailed description of these parameters).

4 Memory Maps

This chapter gives an overview of the TC23x and TC22x memory maps, and describes the address locations and access possibilities for the units, memories, and reserved areas as “seen” from the two different on-chip buses’ point of view.

The TC23x has the following CPU related memories:

- Program Memory Unit (PMU0) with
  - 2 Mbyte of Program Flash Memory (PFLASH)
  - Data Flash Memory (DF_EEPROM)
  - User Configuration Blocks (DF_UCB)
  - 32 Kbyte of Boot ROM (BROM)
- CPU0:
  - 8 Kbyte of Program Scratch-Pad SRAM (PSPR)[1)]
  - 184 Kbyte of Data Scratch-Pad SRAM (DSPR)[1)]
  - 8 Kbyte of Program Cache (PCache)
- LMU:
  - 0 Kbyte SRAM (LMURAM)[1)]

The TC22x has the following CPU related memories:

- Program Memory Unit (PMU0) with
  - 1 Mbyte of Program Flash Memory (PFLASH)
  - Data Flash Memory (DF_EEPROM)
  - User Configuration Blocks (DF_UCB)
  - 32 Kbyte of Boot ROM (BROM)
- CPU0:
  - 8 Kbyte of Program Scratch-Pad SRAM (PSPR)[1)]
  - 88 Kbyte of Data Scratch-Pad SRAM (DSPR)[1)]
  - 8 Kbyte of Program Cache (PCache)
- LMU:
  - 0 Kbyte SRAM (LMURAM)[1)]

Furthermore, the each device has two on-chip buses:

- System Peripheral Bus (SPB)
- Shared Resource Interconnect (SRI)

1) Before used by the application, the memory has to be initialized (see chapter ´Memory Test Unit´, MTU)

4.1 How to Read the Address Maps

The bus-specific address maps describe how the different bus master devices react on accesses to on-chip memories and modules, and which address ranges are valid or invalid for the corresponding buses.

The detailed address mapping of e.g. control registers, sram blocks or flash banks/sectors within a module is described in the related module chapter.

The SPB Bus address map shows the system addresses from the point of view of the SPB master agents[1)].

The SRI address map shows the system addresses from the point of view of the SRI master agents[1)].

The SFI is a bi-directional bridge between SRI and SPB and therefore not mentioned here as SRI or SPB master in the Address Map. The SFI is fully transparent and does not include an address translation mechanism.

Note: In addition to the here described system address map, each TriCore has a TriCore IP internal access to its PSPR via C000_0000 and an internal access to its DSPR via D000_0000. This additional/private view to the local scratch pad srams is described in the CPU chapter.

1) Available SRI / SPB master agents are described in the On Chip Bus System chapter

Table 4-1 defines the acronyms and other terms that are used in the address maps (Table 4-2 and Table 4-3).

Table 4-1 Definition of Acronyms and Terms

Term	Description
…BE	Means “Bus error” generation.
SPBBE	A bus access is terminated with a bus error on the SPB.
SRIBE	A bus access is terminated with a bus error on the SRI.
access	A bus access is allowed and is executed.

4.2 Contents of the Segments

This section summarizes the contents of the segments.

The CPU attributes (cached / non-cached) of these segments can be configured for each CPUs data and program side individually (see CPU chapter: Physical Memory Attribute Registers, PMAx).

Segments 0-6

These memory segments are reserved.

Segment 7

This memory segment allows access to the CPUs Program and Data Scratch Pad SRAM (PSPR, DSPR), Program Cache SRAM (PCache) and to the TAG SRAM related to the Program Cache (PTAG SRAM[1)]).

PCache and PTAG SRAMs[1)] can be only accessed if the related Program Cache is disabled[2)].

CPU default attribute for this segment: non-cached.

Segment 8

This memory segment allows access to PFlash and BROM.

The attribute of segments (cached / non-cached) can be configured for each CPUs data and program side individually (see CPU chapter: Physical Memory Attribute Registers, PMAx).

CPU default attributes for this segment: cached.

Segment 9

This memory segment allows access to LMU SRAM (if implemented) and to the EMEM (Emulation Device only).

CPU default attributes for this segment: cached.

Segment 10

This memory segment allows access to PFlash, DFlash and BROM.

CPU default attributes for this segment: non-cached.

1) TAG SRAMs are not meant to be used as general SRAMs and can be accessed only with single data access and only with 64 bit aligned address.

2) Mapping of Cache and TAG SRAMs is controlled via the MTU register MTU_MEMMAP.

Segment 11

This memory segment allows access to LMU SRAM (if implemented) and to the EMEM (Emulation Device only).

CPU default attributes for this segment: non-cached.

Segment 12

This memory segment is reserved.

Segment 13

This memory segment is reserved.

Segment 14

This memory segment is reserved.

Segment 15

This memory segment allows accesses to all SFRs, CSFRs.

Access from DMA Move Engine, Cerberus to the lower 128 MB of this segment are processed by the DMA FPI master interface on the SPB Bus, to the upper 128 MB of this segment by the DMA SRI master interface on the SRI Bus.

Data access from TC1.6 CPUs to the lower 128 MB of this segment are processed by the CPU.DMI FPI master interface on the SPB Bus, to the upper 128 MB of this segment by the CPU.DMI SRI master interface on the SRI Bus.

CPUx attributes for this segment: non-cached.

4.3 Address Map of the On Chip Bus System

This chapter describes the system address map as it is seen from the SRI and SPB bus masters (bus master agents are described in the chapter On Chip Bus System).

All bus master agents can address identical peripherals and memories at identical address. The system address map is visible and valid for all CPUs which means that all peripherals and resources are accessible from all TriCore CPUs and other on chip bus master agents.

Parallel access by more than one bus master agents to one slave agent are executed sequentially. Additionally the SRI and SPB bus do support atomic Read Modify Write sequences from the CPUs. Hardware semaphores for temporary exclusive bus access from one master to a slave are not implemented

4.3.1 Segments 0 to 14

Table 4-2 shows the address map of segments 0 to 14.

Table 4-2 On Chip Bus Address Map of Segment 0 to 14

Segment	Address Range	Size	Description	Access Type	
				Read[1)]	Write[2)]
0-6	$0000\ 0000_H$ - $0000\ 0007_H$	8 byte	Reserved (virtual address space)	SRIBE	SRIBE
	$0000\ 0008_H$ - $6FFF\ FFFF_H$	-		SRIBE	SRIBE
7	$7000\ 0000_H$ - $7002\ DFFF_H$	184 Kbyte	TC23x: CPU0 Data Scratch-Pad SRAM (CPU0.DSPR)	access	access
	$7002\ E000_H$ - $700F\ FFFF_H$	-	TC23x: Reserved	SRIBE	SRIBE
	$7000\ 0000_H$ - $7001\ 5FFF_H$	88 Kbyte	TC22x: CPU0 Data Scratch-Pad SRAM (CPU0.DSPR)	SRIBE	SRIBE
	$7001\ 6000_H$ - $700F\ FFFF_H$	-	TC22x: Reserved	SRIBE	SRIBE
	$7010\ 0000_H$ - $7010\ 1FFF_H$	8 Kbyte	CPU0 Program Scratch-Pad SRAM (CPU0.PSPR)	access	access
	$7010\ 2000_H$ - $7010\ 3FFF_H$	8 Kbyte	CPU0.Program Cache SRAM (CPU0.PCache)	access[3)] / SRIBE	access[3)] / SRIBE

Table 4-2 On Chip Bus Address Map of Segment 0 to 14 (cont'd)

Segment	Address Range	Size	Description	Access Type	
				Read[1)]	**Write[2)]**
	$7010\ 4000_H$ - $701B\ FFFF_H$	-	Reserved	SRIBE	SRIBE
	$701C\ 0000_H$ - $701C\ 0BFF_H$	-	CPU0 Program Cache TAG SRAM[4)] (CPU0.PTAG)	access[3)] / SRIBE	access[3)] / SRIBE
	$701C\ 0C00_H$ - $7FFF\ FFFF_H$	-	Reserved	SRIBE	SRIBE
8	$8000\ 0000_H$ - $801F\ FFFF_H$	2 Mbyte	TC23x: Program Flash (PFlash)	access	access[2)]
	$8020\ 0000_H$ - $8FE6\ FFFF_H$	-	TC23x: Reserved	SRIBE	SRIBE
	$8000\ 0000_H$ - $800F\ FFFF_H$	1 Mbyte	TC22x: Program Flash (PFlash)	access	access[2)]
	$8010\ 0000_H$ - $8FE6\ FFFF_H$	-	TC22x: Reserved	SRIBE	SRIBE
	$8FE7\ 0000_H$ - $8FE7\ 7FFF_H$	32 Kbyte	Online Data Acquisition (OLDA)	SRIBE	access[5)] / SRIBE
	$8FE7\ 8000_H$ - $8FFF\ 7FFF_H$	-	Reserved	SRIBE	SRIBE
	$8FFF\ 8000_H$ - $8FFF\ FFFF_H$	32 Kbyte	Boot ROM (BROM)	access	SRIBE
9	$9000\ 0000_H$ - $9000\ 7FFF_H$	32 Kbyte	TC23x-ADAS: LMU SRAM (LMURAM)	access	access
	$9000\ 0000_H$ - $9000\ 7FFF_H$	-	TC23x, TC22x: Reserved	SRIBE	SRIBE
	$9000\ 8000_H$ - $9EFF\ FFFF_H$	-	Reserved	SRIBE	SRIBE
	$9F00\ 0000_H$ - $9F07\ FFFF_H$	512 KByte	TC23x-ADAS: Extended Memory (EMEM)	access	access
	$9F00\ 0000_H$ - $9F07\ FFFF_H$	-	TC23x, TC22x: Reserved	SRIBE	SRIBE
	$9F08\ 0000_H$ - $9F0F\ FFFF_H$	-	Reserved	SRIBE	SRIBE

Table 4-2 On Chip Bus Address Map of Segment 0 to 14 (cont'd)

Segment	Address Range	Size	Description	Access Type	
				Read[1)]	**Write[2)]**
	$9F10\ 0000_H$ - $9F10\ 3FFF_H$	16 Kbyte	TC23x-ADAS: Extended Memory (EMEM)	access	access
	$9F10\ 0000_H$ - $9F10\ 3FFF_H$	-	TC23x, TC22x: Reserved	SRIBE	SRIBE
	$9F10\ 4000_H$ - $9FFF\ FFFF_H$	-	Reserved	SRIBE	SRIBE
10	$A000\ 0000_H$ - $A01F\ FFFF_H$	2 Mbyte	TC23x: Program Flash (PFlash)	access	access[2)]
	$A020\ 0000_H$ - $AEFF\ FFFF_H$	-	TC23x: Reserved	SRIBE	SRIBE
	$A000\ 0000_H$ - $A00F\ FFFF_H$	1 Mbyte	TC22x: Program Flash (PFlash)	access	access[2)]
	$A010\ 0000_H$ - $AEFF\ FFFF_H$	-	TC22x: Reserved	SRIBE	SRIBE
	$AF00\ 0000_H$ - $AF0F\ FFFF_H$	1 Mbyte	Data Flash 0 (DF0 Address Range)	access	access[2)6)]
	$AF10\ 0000_H$ - $AF10\ 3FFF_H$	16 Kbyte	Data Flash 0 (DF0 Address Range)	access	access[2)6)]
	$AF10\ 4000_H$ - $AFE6\ FFFF_H$	-	Reserved	SRIBE	SRIBE
	$AFE7\ 0000_H$ - $AFE7\ 7FFF_H$	32 Kbyte	Online Data Acquisition (OLDA)	SRIBE	access[5)] / SRIBE
	$AFE7\ 8000_H$ - $AFFF\ 7FFF_H$	-	Reserved	SRIBE	SRIBE
	$AFFF\ 8000_H$ - $AFFF\ FFFF_H$	32 Kbyte	Boot ROM (BROM)	access	SRIBE
11	$B000\ 0000_H$ - $B000\ 7FFF_H$	32 Kbyte	TC23x-ADAS: LMU SRAM (LMURAM)	access	access
	$B000\ 0000_H$ - $B000\ 7FFF_H$	-	TC23x, TC22x: Reserved	SRIBE	SRIBE
	$B000\ 8000_H$ - $BDFF\ FFFF_H$	-	Reserved	SRIBE	SRIBE

Table 4-2 On Chip Bus Address Map of Segment 0 to 14 (cont'd)

Segment	Address Range	Size	Description	Access Type	
				Read[1)]	Write[2)]
	BE00 0000$_H$ - BE07 FFFF$_H$	512 Kbyte	TC23x-ADAS: FFT Accelerator	access	access
	BE00 0000$_H$ - BE07 FFFF$_H$	-	TC23x, TC22x: Reserved	SRIBE	SRIBE
	BE08 0000$_H$ - BE0F FFFF$_H$	-	Reserved	SRIBE	SRIBE
	BE10 0000$_H$ - BE17 FFFF$_H$	512 Kbyte	TC23x-ADAS: FFT Accelerator	access	access
	BE10 0000$_H$ - BE17 FFFF$_H$	-	TC23x, TC22x: Reserved	SRIBE	SRIBE
	BE18 0000$_H$ - BEFF FFFF$_H$	-	Reserved	SRIBE	SRIBE
	BF00 0000$_H$ - BF07 FFFF$_H$	512 KByte	TC23x-ADAS: Extended Memory (EMEM)	access	access
	BF00 0000$_H$ - BF07 FFFF$_H$	-	TC23x, TC22x: Reserved	SRIBE	SRIBE
	BF08 0000$_H$ - BF0F FFFF$_H$	-	Reserved	SRIBE	SRIBE
	BF10 0000$_H$ - BF10 3FFF$_H$	16 Kbyte	TC23x-ADAS: Extended Memory (EMEM)	access	access
	BF10 0000$_H$ - BF10 3FFF$_H$	-	TC23x, TC22x: Reserved	SRIBE	SRIBE
	BF10 4000$_H$ - BFFF FFFF$_H$	-	Reserved	SRIBE	SRIBE
12	C000 0000$_H$ - CFFF FFFF$_H$	-	Reserved[7)]	SRIBE	SRIBE
13	D000 0000$_H$ - DFFF FFFF$_H$	-	Reserved[7)]	SRIBE	SRIBE
14	E000 0000$_H$ - EFFF FFFF$_H$	-	Reserved	SRIBE	SRIBE
15	F000 0000$_H$ - FFFF FFFF$_H$	256 Mbyte	see **Table 4-3**		

1) A read transaction through the SRI to FPI bridge that is terminated with Bus Error will result in Bus Errors on SRI and FPI (valid for transactions from FPI to SRI and SRI to FPI).

2) Write access to Flash resources are handled by the PMU module (Flash command sequence, see PMU chapter for details).
3) PCache SRAMs and the corresponding TAG SRAM can be only accessed when mapped into the address space (PCache / DCache disabled. See TAG chapter, register MTU_MEMMAP for details).
4) TAG SRAMs are not meant to be used as general SRAMS and can be accessed only with single data access and only with 64 bit aligned address. Mapping of TAG SRAMs in the address map can be used as additional option for memory testing.
5) Online Data Acquisition address space can be disabled/enabled via LMU control register bit LMU_MEMCON.OLDAEN.
6) DF0 is the address where all Data Flash blocks are mapped to. The details about the Data Flash block sizes, segmentation and exact mapping are described in the chapter PMU.
7) See also chapter ´CPU, ´Local and Global Adressing´ for CPU local views to segment ´C´ and segment ´D´.

4.3.2 Segment 15

Table 4-3 shows the address map of segment ´F´ as seen from the SRI and SPB bus masters (bus master agents are described in the chapter On Chip Bus System).

Table 4-1 gives an overview about the address mapping of the module address ranges:

- which modules are mapped into the first 16 KB of segment ´F´ and can be accessed by the TC1.6E/P with absolute addressing modes (left side)
- examples of covering modules with relative addressing mode (base address +- 32 KB, in the middle)

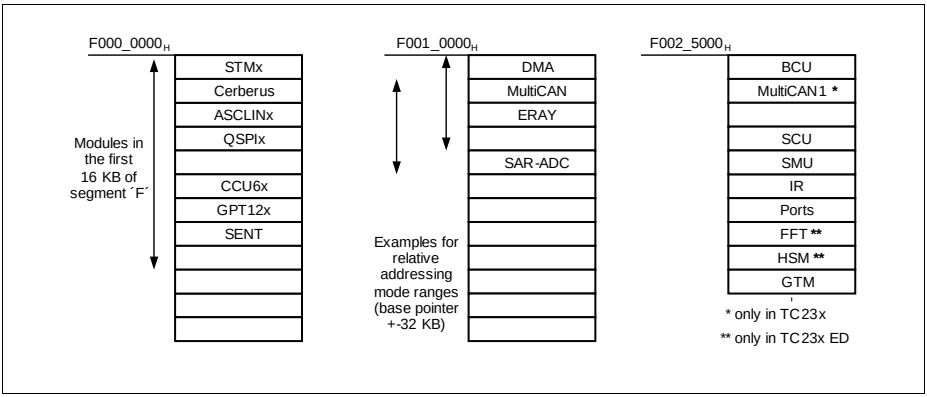


Figure 4-1 Segment F Structure

Please note that **Table 4-3** describes the mapping of modules to segment F. The details of the module address ranges can be found in the module chapters register overview.

Table 4-3 On Chip Bus Address Map of Segment 15

Unit	Address Range	Size	Access Type	
			Read	Write
System Timer 0 (STM0)	$F000\ 0000_H$ - $F000\ 00FF_H$	256 byte	access	access
Reserved	$F000\ 0100_H$ - $F000\ 03FF_H$	–	SPBBE	SPBBE
On-Chip Debug Support (Cerberus)	$F000\ 0400_H$ - $F000\ 05FF_H$	2x256 byte	access	access
ASCLIN0 (ASCLIN0)	$F000\ 0600_H$ - $F000\ 06FF_H$	256 byte	access	access

Table 4-3 On Chip Bus Address Map of Segment 15 (cont'd)

Unit	Address Range	Size	Access Type	
			Read	Write
ASCLIN1 (ASCLIN1)	$F000\ 0700_H$ - $F000\ 07FF_H$	256 byte	access	access
Reserved	$F000\ 0800_H$ - $F000\ 1BFF_H$	–	SPBBE	SPBBE
QUEUED SPI 0 (QSPI0)	$F000\ 1C00_H$ - $F000\ 1CFF_H$	256 byte	access	access
QUEUED SPI 1 (QSPI1)	$F000\ 1D00_H$ - $F000\ 1DFF_H$	256 byte	access	access
QUEUED SPI 2 (QSPI2)	$F000\ 1E00_H$ - $F000\ 1EFF_H$	256 byte	access	access
QUEUED SPI 3 (QSPI3)	$F000\ 1F00_H$ - $F000\ 1FFF_H$	256 byte	access	access
Reserved	$F000\ 2000_H$ - $F000\ 29FF_H$	–	SPBBE	SPBBE
Capture/Compare Unit 6 0(CCU60)	$F000\ 2A00_H$ - $F000\ 2AFF_H$	256 byte	access	access
Capture/Compare Unit 6 1 (CCU61)	$F000\ 2B00_H$ - $F000\ 2BFF_H$	256 byte	access	access
Reserved	$F000\ 2C00_H$ - $F000\ 2DFF_H$	–	SPBBE	SPBBE
General Purpose Timer 12 0 (GPT120)	$F000\ 2E00_H$ - $F000\ 2EFF_H$	256 byte	access	access
Reserved	$F000\ 2F00_H$ - $F000\ 2FFF_H$	–	SPBBE	SPBBE
SENT Module (SENT)	$F000\ 3000_H$ - $F000\ 3AFF_H$	11x256 byte	access	access
Reserved	$F000\ 3B00_H$ - $F000\ FFFF_H$	–	SPBBE	SPBBE
Direct Memory Access Controller (DMA)	$F001\ 0000_H$ - $F001\ 3FFF_H$	16 KByte	access	access
Reserved	$F001\ 4000_H$ - $F001\ 7FFF_H$	–	SPBBE	SPBBE

Table 4-3 On Chip Bus Address Map of Segment 15 (cont'd)

Unit	Address Range	Size	Access Type	
			Read	Write
MultiCAN Controller (CAN)	F001 8000$_H$ - F001 BFFF$_H$	16 Kbyte	access	access
TC22x: Reserved	F001 C000$_H$ - F001 FFFF$_H$	–	SPBBE	SPBBE
TC23x: FlexRay™ Protocol Controller (E-Ray)	F001 C000$_H$ - F001 CFFF$_H$	4 Kbyte	access	access
TC22x, TC23x: Reserved	F001 D000$_H$ - F001 FFFF$_H$	–	SPBBE	SPBBE
TC23x-ADAS: Ethernet Controller System Control Register (ETH)	F001 D000$_H$ - F001 D0FF$_H$	256 byte	access	access
TC23x-ADAS: Reserved	F001 D100$_H$ - F001 DFFF$_H$	–	SPBBE	SPBBE
TC23x-ADAS: Ethernet Controller (ETH)	F001 E000$_H$ - F001 FFFF$_H$	8 KByte	access	access
Analog-to-Digital Converter (VADC)	F002 0000$_H$ - F002 3FFF$_H$	16 KByte	access	access
Reserved	F002 4000$_H$ - F002 7FFF$_H$	–	access	access
TC23x: MultiCAN1 Controller (MultiCAN1)	F002 8000$_H$ - F002 BFFF$_H$	16 Kbyte	access	access
TC22x: Reserved	F002 8000$_H$ - F002 BFFF$_H$	–	SPBBE	SPBBE
Reserved	F002 C000$_H$ - F002 FFFF$_H$	–	SPBBE	SPBBE
System Peripheral Bus Control Unit (BCU)	F003 0000$_H$ - F003 00FF$_H$	256 byte	access	access
Reserved	F003 0100$_H$ - F003 4FFF$_H$	–	SPBBE	SPBBE
I/O Monitor (IOM)	F003 5000$_H$ - F003 51FF$_H$	2x256 byte	access	access
Reserved	F003 5200$_H$ - F003 5FFF$_H$	–	SPBBE	SPBBE

Table 4-3 On Chip Bus Address Map of Segment 15 (cont'd)

Unit	Address Range	Size	Access Type	
			Read	Write
System Control Unit (SCU)	F003 6000$_H$ - F003 63FF$_H$	1 Kbyte	access	access
Reserved	F003 6400$_H$ - F003 67FF$_H$	–	SPBBE	SPBBE
Safety Management Unit (SMU)	F003 6800$_H$ - F003 6FFF$_H$	2 Kbyte	access	access
Interrupt Router (IR)	F003 7000$_H$ - F003 7FFF$_H$	4 Kbyte	access	access
Interrupt Router (IR) SRC Registers	F003 8000$_H$ - F003 9FFF$_H$	8 Kbyte	access	access
Port 00	F003 A000$_H$ - F003 A0FF$_H$	256 byte	access	access
Reserved	F003 A100$_H$ - F003 A1FF$_H$	256 byte	SPBBE	SPBBE
Port 02	F003 A200$_H$ - F003 A2FF$_H$	256 byte	access	access
Reserved	F003 A300$_H$ - F003 AFFF$_H$	–	SPBBE	SPBBE
Port 10	F003 B000$_H$ - F003 B0FF$_H$	256 byte	access	access
Port 11	F003 B100$_H$ - F003 B1FF$_H$	256 byte	access	access
Reserved	F003 B200$_H$ - F003 B2FF$_H$	–	SPBBE	SPBBE
Port 13	F003 B300$_H$ - F003 B3FF$_H$	256 byte	access	access
Port 14	F003 B400$_H$ - F003 B4FF$_H$	256 byte	access	access
Port 15	F003 B500$_H$ - F003 B5FF$_H$	256 byte	access	access
Reserved	F003 B600$_H$ - F003 BFFF$_H$	–	SPBBE	SPBBE

Table 4-3 On Chip Bus Address Map of Segment 15 (cont'd)

Unit	Address Range	Size	Access Type	
			Read	Write
Port 20	F003 C000$_H$ - F003 C0FF$_H$	256 byte	access	access
Port 21	F003 C100$_H$ - F003 C1FF$_H$	256 byte	access	access
Port 22	F003 C200$_H$ - F003 C2FF$_H$	256 byte	access	access
Port 23	F003 C300$_H$ - F003 C3FF$_H$	256 byte	access	access
Reserved	F003 C400$_H$ - F003 D1FF$_H$	–	SPBBE	SPBBE
Reserved	F003 D200$_H$ - F003 D2FF$_H$	–	SPBBE	SPBBE
Port 33	F003 D300$_H$ - F003 D3FF$_H$	256 byte	access	access
Port 34	F003 D400$_H$ - F003 D4FF$_H$	256 byte	access	access
Reserved	F003 D500$_H$ - F003 DFFF$_H$	–	SPBBE	SPBBE
Port 40	F003 E000$_H$ - F003 E0FF$_H$	256 byte	access	access
Port 41	F003 E100$_H$ - F003 E1FF$_H$	256 byte	access	access
Reserved	F003 E200$_H$ - F003 FFFF$_H$	–	SPBBE	SPBBE
TC23x: High Security Module (HSM)	F004 0000$_H$ - F005 FFFF$_H$	128 Kbyte	access	access
TC22x: Reserved	F004 0000$_H$ - F005 FFFF$_H$	–	SPBBE	SPBBE
Memory Test Unit (MTU)	F006 0000$_H$ - F006 FFFF$_H$	64 Kbyte	access	access
Reserved	F007 0000$_H$ - F00F FFFF$_H$	–	SPBBE	SPBBE

Table 4-3 On Chip Bus Address Map of Segment 15 (cont'd)

Unit	Address Range	Size	Access Type	
			Read	Write
Global Timer Module (GTM)	$F010\ 0000_H$ - $F019\ FFFF_H$	640 Kbyte	access	access
Reserved	$F01A\ 0000_H$ - $F7FF\ FFFF_H$	–	SPBBE	SPBBE
Reserved	$F800\ 0000_H$ - $F800\ 04FF_H$	–	SRIBE	SRIBE
Program Memory Unit 0 (PMU0)	$F800\ 0500_H$ - $F800\ 05FF_H$	256 byte	access	access
Reserved	$F800\ 0600_H$ - $F800\ 0FFF_H$	–	SRIBE	SRIBE
Flash Register (PMU0)	$F800\ 1000_H$ - $F800\ 23FF_H$	5 Kbyte	access	access
Reserved	$F800\ 2400_H$ - $F86F\ FFFF_H$	–	SRIBE	SRIBE
SRI Crossbar (XBar_SRI)	$F870\ 0000_H$ - $F870\ 04FF_H$	5x256 byte	access	access
Reserved	$F870\ 0500_H$ - $F870\ 07FF_H$	–	SRIBE	SRIBE
Local Memory Unit (LMU)	$F870\ 0800_H$ - $F870\ 08FF_H$	256 byte	access	access
Reserved	$F870\ 0900_H$ - $F870\ 0BFF_H$	–	SRIBE	SRIBE
TC23x-ADAS: FFT Accelerator (FFT, Emulation Device only)	$F870\ 0C00_H$ - $F870\ 0CFF_H$	256 byte	access	access
TC23x, TC22x: Reserved	$F870\ 0C00_H$ - $F870\ 0CFF_H$	–	SRIBE	SRIBE
Reserved	$F870\ 0D00_H$ - $F87F\ FFFF_H$	–	SRIBE	SRIBE
CPU0 SFR	$F880\ 0000_H$ - $F880\ FFFF_H$	64 KByte	access	access
CPU0 CSFR	$F881\ 0000_H$ - $F881\ FFFF_H$	64 KByte	access	access

Table 4-3 On Chip Bus Address Map of Segment 15 (cont'd)

Unit	Address Range	Size	Access Type	
			Read	Write
Reserved	$F882\ 0000_H$ - $F8FF\ FFFF_H$	–	SRIBE	SRIBE
TC23x-ED: Reserved for Emulation Device Registers (ED Reg)	$F900\ 0000_H$ - $F90F\ FFFF_H$	1 MB	access	access
TC23x, TC23x-ADAS, TC22x: Reserved	$F900\ 0000_H$ - $F90F\ FFFF_H$	–	SRIBE	SRIBE
Reserved	$F910\ 0000_H$ - $FFFF\ FFFF_H$	–	SRIBE	SRIBE

4.4 Memory Module Access Restrictions

Table 4-4 describes which type of accesses are possible to the different memories in the TC21x/TC22x/TC23x.

Table 4-4 Possible Memory Accesses[1)]

Memory		Bit	Byte		Half-word		Word		Double-word	
		rmw	r	w	r	w	r	w	r	w
PMI[2)]	PSPR	y	y	y	y	y	y	y	y	y
	PTAG[3)]	-	-	-	-	-	y	y	-	-
	PCACHE	y	y	y	y	y	y	y	y	y
DMI[2)]	DSPR	y	y	y	y	y	y	y	y	y
LMU[2)]	LMURAM	y	y	y	y	y	y	y	y	y
PMU	BROM	-	y	-	y	-	y	-	y	-
	PFLASH	-	y	-	y	-	y	y	y	y
	DFLASH	-	y	-	y	-	y	y	y	y

1) ´y´ means: supported. ´-´ means: not supported

2) The module also supports SRI 2-Word and 4-Word Block read and write accesses.

3) TAG SRAMs are not meant to be used as general SRAMS and can be accessed only with 32-bit data access and only with 64 bit aligned address. Mapping of TAG SRAMs in the address map can be used as additional option for memory testing. The TAG SRAM size is below 24 bit, so the 8 MSB of a 32 bit write or read are dont care.

4.5 Side Effects from Modules to CPU0 Data Scratch Pad SRAM (DSPR0)

Pls. note that a part of the CPU0 DSPR will be overwritten by the start-up (BootROM) procedure after cold power-on as well as the information in this area should be used and in case modified by the (user) SW procedure when preparing the device to enter Stand-By mode. The details are described in the Firmware chapter, sub-chapter ´RAMs Handling´.

5 TC21x/TC22x/TC23x BootROM Content

The TC21x/TC22x/TC23x BootROM consists basically of three parts:

- Startup Software (short name SSW);
- Software modules implementing additional functions (Bootstrap Loaders);
- Test Firmware.

5.1 Startup Software

The Startup Software is the first software executed after a chip reset.

SSW is executed on CPU0 - all other CPUs are kept in Halt-state during boot, to be started by user software whereas:

- SSW start address in BootROM is the reset value in Program Counter register of the CPU0. From this location an instruction is fetched and this is the first instruction executed after any device start-up.
  - immediately after this entry point the firmware checks for testmode, and in case testmode is selected - jump to test firmware is executed
- the last SSW instruction performs a jump to the first user code instruction. This first user instruction can be fetched from different locations depending on the start-up configuration as selected by the user.

The Startup Software contains procedures to initialize the device depending on one or more from the following:

- information previously stored into dedicated Flash locations
- the current state of special bits/fields in dedicated register/memory locations
- the type of event which has triggered the SSW-execution (the last reset event)
- values applied to external (configuration-) pins (optional)

The SSW also calls - in case - other firmware modules.

5.1.1 Events triggering SSW execution

SSW execution can be triggered by different events. SSW recognizes the triggering (reset) event and takes (partially) different execution flows.

5.1.1.1 Power-on

This is the initial powering-up of the device after the supply has been switched off, or in other words - the only way to generate this reset event is by applying power to a previously un-powered device.

The conditions at which the SSW execution starts upon this event include:

- all the registers are in their initial (reset) state
- Flash is under reset - meaning not active, and not ready to perform any (read/erase/program) operation

- RAMs' content is undefined
- clock system is in its initial state

Due to the overall "fully initial" state of the device upon power-on, the SSW flow is respectively longer in this case and covers the biggest amount of activities compared to other reset events.

5.1.1.2 System reset

Attention: From SSW point of view, the handling of system reset is generally the same as of "warm power-on".
Therefore further in this Chapter when system reset is referred - the same SSW handling applies upon warm power-on, exceptions will be in case specially notified.

This reset event can be requested by different sources:

- device internal hardware - from modules like watchdog timer and security/memory control logic
- external hardware - when active signal is applied to defined device pin(s)
- software - when defined control bit(s) are respectively installed during user code execution

For most of the sources, generating system reset is a feature configurable by software.

The exception is PORST pin, applying active low level to which generates system reset only if the supply voltage is above defined level permanently in the surrounding time window - e.g. the device is continuously powered before and during system reset activation. Otherwise - upon a supply drop below some level - power-on is immediately triggered. All this functionality is supported by the EVR module.

The conditions at which SSW execution starts upon system reset include:

- all the registers are in their initial (reset) state
- Flash is under reset - meaning not active, and not ready to perform any (read/erase/program) operation
- RAMs' content is the same as just before the system reset has been triggered
- clock system is in its initial state

As seen from the above list, the only difference in the initial system status after power-on and system reset is in that the content of RAMs is not changed by the reset event.

5.1.1.3 Application reset

Similarly to system reset, an application reset can be requested by different sources: internal/external hardware as well as software. For all the sources, generating system reset is a feature configurable by software.

The conditions at which SSW execution starts upon application reset include:

- registers connected to this reset type are in their initial (reset) state

- Flash is in read mode
- all the rest - RAMs and surround logic, clock system, registers under system reset - is not affected by this event.

Following the limited changes within the device status upon application reset, the SSW flow is shortest in this case.

5.1.2 Clock system during start-up

The state of clock system during device start-up depends on the reset event type:

- upon power-on and system reset - clock system is in its initial state, namely:
  - f_{SRI}= f_{CPU0} = f_{FSI}=f_{BACK}=100MHz nominal
  - f_{SPB}=f_{STM}=f_{BACK}/2=50MHz nominal
  - in Emulation Device (ED) f_{BBB}=f_{BACK}/2=50MHz nominal
  - PLL and VCO are in power-down mode
- upon application reset - clock system does not change its state, therefore the device runs at same frequency and clock source as before the reset event

With the same clock system state the first user code is started, with exception of the Bootstrap Loader mode upon power-on - refer to **Chapter 5.2**.

5.1.3 RAM overwrite during start-up

Start-up procedure can overwrite up to 8kByte at the beginning of CPU0 DSPR. Therefore this area should not be used by application software to save data, which must be preserved through device resets.

Additionally, SSW performs a special handling of CPU0 DSPR upon exit from stand-by mode if this RAM has been kept supplied during stand-by. To assure this handling will be correct:

- upon cold power-on reset, SSW stores 16 Words (total of 64 Bytes) information into so-called “reserved area” in CPU0_DSPR starting at address $D000'2000_H$
- if the application wants to keep CPU0 DSPR supplied during stand-by mode and to save information there:
  - before going into stand-by mode, the user code must execute CPU0 DSPR preparation as described in **Chapter 5.4**
  - the user code must not touch the data within reserved area (see above) at $D000'2000_H$...$D000'203F_H$ except when executing **Preparation before to enter Stand-by mode**

5.1.4 Boot Options Summary

This chapter summarizes the TC23x startup configurations in user mode.

Internal Start

In this basic startup mode, the first user instruction is fetched from address A000 0020_H in Internal Program Flash of the device.

Bootloader Modes

Different Bootstrap Loader routines are used in these modes to download code/data into the Instruction Scratchpad Memory SPRAM (PSPR), as follows:

- ASC bootloader
- CAN bootloader
- Generic bootloader mode - the bootloading procedure itself here is in fact one of the two above possibilities - ASC/CAN. The selection either to use ASC- or CAN-communication protocol is done by the SSW upon the first byte received at the couple of pins, which are then respectively configured to ASC or to MultiCAN module.

Alternate Boot Modes

In these modes, program code is started from user-defined address but only if all defined check-conditions are satisfied. Otherwise Generic Bootstrap Loader mode can be entered to download the code into device, this code is afterwards started by the SSW.

All the information needed for the SSW to handle ABM startup mode is collected into the so-called Headers. The checks are performed according to **Start-up mode selection** flow as defined in **Chapter 5.1.5**.

5.1.5 Start-up mode selection

TC21x/TC22x/TC23x start-up flow in regard to mode selection is shown at **Figure 5-1**.

There is one way only to select start-up configuration in TC21x/TC22x/TC23x:

- **Configuration by Boot Mode Index (BMI)** - according to values taken from dedicated locations in Flash (new feature for AURIX products)

Additionally, as an option of the BMI-configuration flow the well known proceeding from previous devices is supported:

- **Hardware configuration** - according to the values at configuration pins

The flow to evaluate start-up configuration in TC21x/TC22x/TC23x is as follows:

1. if error happens during Flash ramp-up or Flash Config sector is corrupted (unrecoverable data error) - SSW terminates immediately
2. SSW evaluates sequentially up to four Boot Mode Headers BMHD0...BMHD3 - for header locations refer to **Table 5-2**, evaluation procedure - see below and **Figure 5-2**
   a) if evaluation procedure is exited with OK - valid BMI and (in case) code found, start-up mode is taken accordingly
   b) if evaluation procedure is exited with FAIL - continue with step 3.
3. check the firmware internal flag CFGPIN:

a) if set - meaning valid BMI enabling pin-configuration was found, ABM was selected from pins but code check failed (refer to **Figure 5-2**): re-install STSTAT.HWCFG from ABM to BSL (Generic/ASC) accordingly and take the respective start-up mode
b) if not set - no valid BMI and code are found, continue with step 4.

4. check if the condition (HSM boot disabled) AND (OTP protected mode disabled) is true:
   a) if yes - initialize security relevant RAMs, then check either debugger is connected and halt-after-reset is requested (OSTATE.HARR=1)
      - if yes - execute Internal start mode - CPU0 will be halted before the first user instruction
      - if not - execute Generic Bootstrap Loader mode - default mode if no valid BMI
   b) if not - check and in case enable debug access using the general evaluation sequence, then enter endless loop - HSM or an external debugger (if access granted) are able to handle further the device start-up

Note: The initialization of security relevant RAMs here is to prevent using BSL mode as back door for reading out user information previously stored into some RAM(s).

Table 5-1 Boot Mode Header (BMHD) structure

Offset Addr.	Size Byte	Field Name	Description
00_H	4	STADABM	User Code Start Address
04_H	2	BMI [1]	Boot Mode Index (BMI)
06_H	2	BMHDID [1]	Boot Mode Header ID (Confirmation code) = $B359_H$
08_H	4	ChkStart	Memory Range to be checked - Start Address
$0C_H$	4	ChkEnd	Memory Range to be checked - End Address
10_H	4	CRCrange	Check Result [2] for the Memory Range
14_H	4	$\overline{\text{CRCrange}}$	Inverted Check Result for the Memory Range
18_H	4	CRChead	Check Result [2] for the ABM Header (offset 00_H..17_H)
$1C_H$	4	$\overline{\text{CRChead}}$	Inverted Check Result for the ABM Header

1) These fields are different from the ABM Headers in previous devices.

2) CRC calculation is based on IEEE 802.3, the CRC32 ethernet polynomial used is $04C11DB71_H$

Table 5-2 Boot Mode Headers locations

Header	Start address	End address
BMHD0	$A000\ 0000_H$	$A000\ 001F_H$
BMHD1	$A002\ 0000_H$	$A002\ 001F_H$

Table 5-2 Boot Mode Headers locations

Header	Start address	End address
BMHD2	A000 FFE0$_H$	A000 FFFF$_H$
BMHD3	A001 FFE0$_H$	A001 FFFF$_H$

Table 5-1 shows the Boot Mode Header (BMHD) structure, the procedure to evaluate one BMHD includes following steps (refer to **Figure 5-2**):

1. check either Boot Mode Header ID is correct
   a) if BMHDID = B359$_H$ - continue with 2.
   b) if not - continue with 8.
2. calculate the CRC of the first 24 Bytes from the ABM Header (refer to **Table 5-1**) - process the fields STADABM...CRCRange at offsets 00$_H$...17$_H$
   a) compare the result with the CRChead value (offset 18$_H$)
      - if OK - continue with 2.b)
      - if not - continue with 8.
   b) inverse the result value and compare with $\overline{\text{CRChead}}$ (offset 1C$_H$)
      - if OK - continue with 3.
      - if not - continue with 8.
3. check either BMI[15:10, 7:0] value is valid - refer to **Chapter 5.1.5.2**
   a) if yes - continue with 4.
   b) if not - continue with 8.

Note: Bits BMI[9:8] are excluded from this check because they control Lockstep Logic and can have arbitrary values.

4. check the conditions BMI[3]=0 (pin-configuration enabled) AND HWCFG[3] pin value=0 (pin-configuration selected) AND PROCONOTP[30:29]=00$_B$ (OTP protected mode not enabled):
   a) if all the conditions are true - **Hardware configuration** is taken (refer to **Chapter 5.1.5.1** and respective value installed in STSTAT.HWCFG, then continue with 5.
   b) if not - **Configuration by Boot Mode Index (BMI)** is taken (refer to **Chapter 5.1.5.2**) and respective value installed in STSTAT.HWCFG, then continue with 5.
5. check either STSTAT.HWCFG value selects Alternate Boot Mode (ABM)
   a) if yes - continue with 6.
   b) if not - exit this procedure, BMI is OK, mode is selected in STSTAT.HWCFG
6. calculate the CRC over the memory address range ChkStart...ChkEnd (start- and end- addresses taken from offsets 08$_H$ and 0C$_H$ respectively)
   a) compare the result with the CRCrange value (offset 10$_H$)
      - if OK - continue with 6.b)
      - if not - continue with 7.

b) inverse the result value and compare with $\overline{\text{CRCrange}}$ (offset 14_H)
- if OK - exit this procedure, BMI and code are OK, ABM is selected in STSTAT.HWCFG
- if not - continue with 7.

7. check how is ABM selected
   a) from BMI value - continue with 8.
   b) from HWCFG pins - install Generic Bootstrap Loader ABM in STSTAT.HWCFG, exit this procedure with OK
8. check again so called Magic number in Flash Config sector
   a) if OK - exit this procedure, header and BMI check failed
   b) if wrong - terminate SSW immediately on error

Note: The last step above is to reduce the probability that a hacker attack disturbs BMI/ABM checks targeting to cause unintentional (from the original user code) entry into Bootstrap Loader mode.

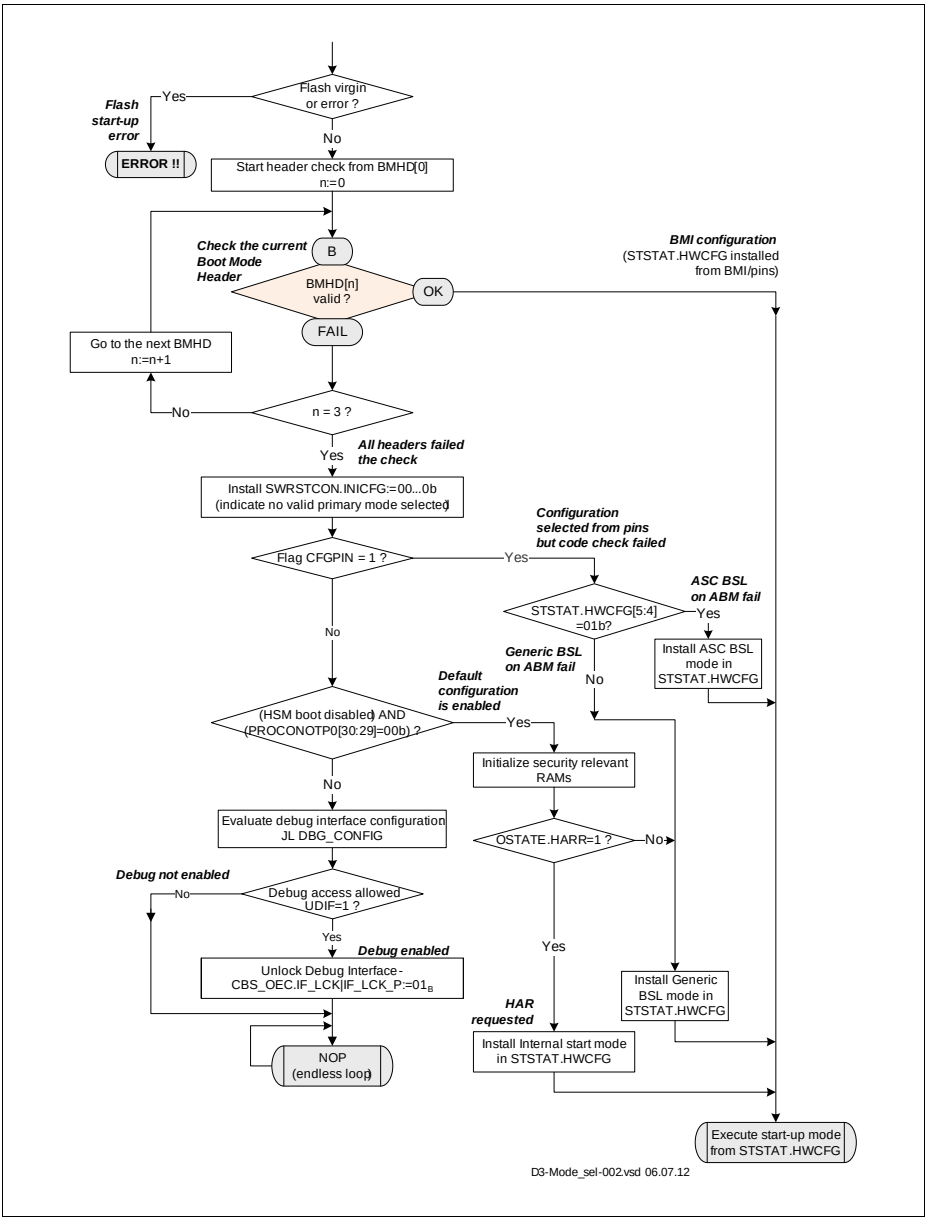


Figure 5-1 Start-up mode selection flow in TC21x/TC22x/TC23x

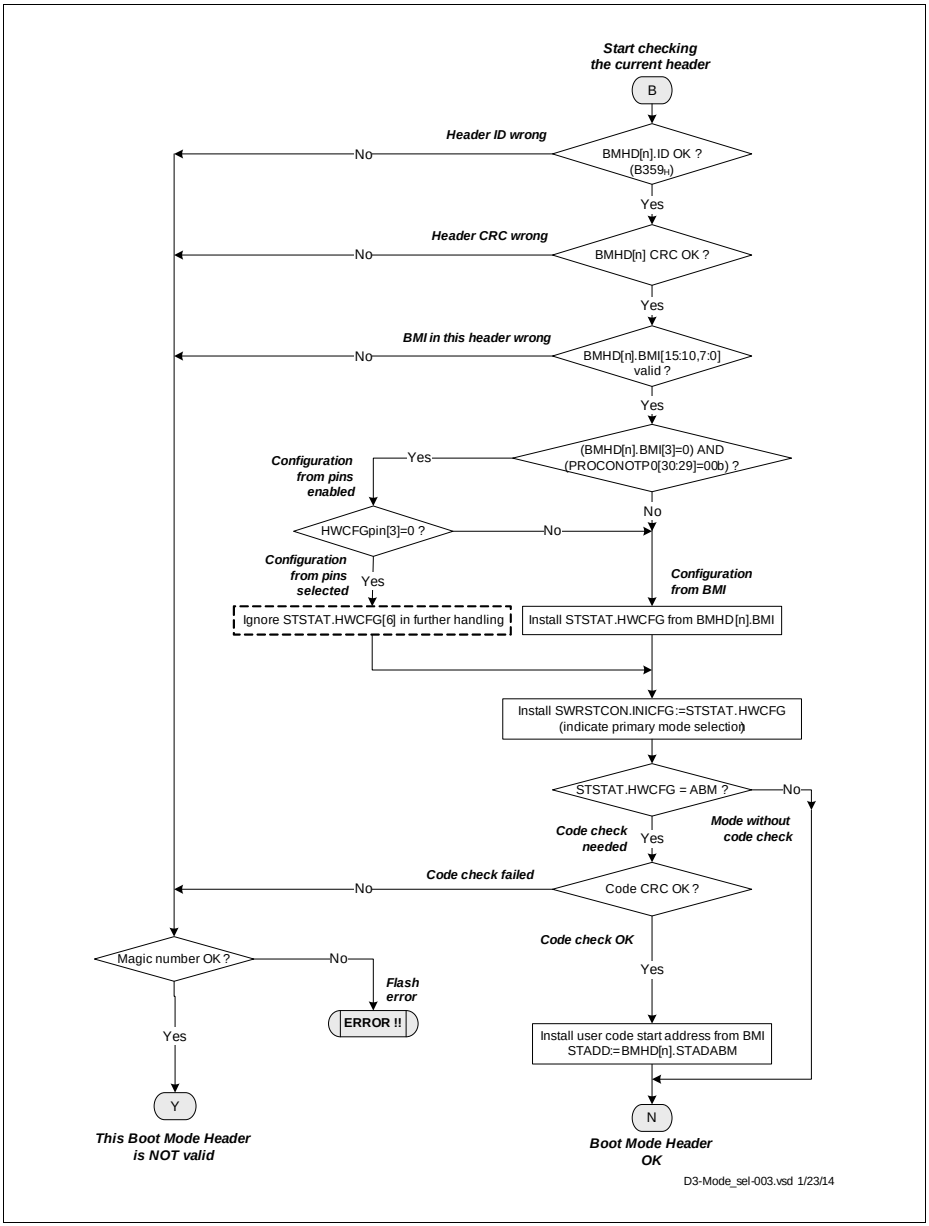


Figure 5-2 Mode selection in TC21x/TC22x/TC23x: check of one Boot Mode

Header

5.1.5.1 Hardware configuration

The start-up mode selection by configuration pins HWCFG[5:4] in TC21x/TC22x/TC23x is shown in **Table 5-3**. The values at these pins are latched by hardware upon any reset (deactivation - rising edge) into register which is read and evaluated by SSW during start-up processing.

Note: HWCFG[2:0] pins are used for EVR control, HWCFG[6] - for pull-up/tristate pin configuration,HWCFG[7] is not available for TC21x/TC22x/TC23x.

Table 5-3 Start-up mode selection by pins in TC21x/TC22x/TC23x

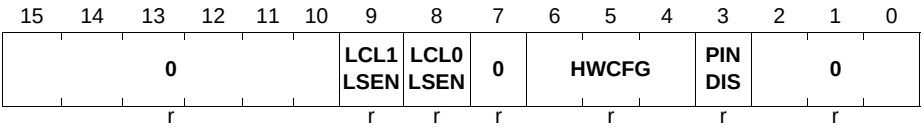
HWCFG pins		Start-up Mode
[5]	[4]	
1	1	Internal start from Flash
1	0	ABM, Generic Bootstrap Loader on fail
0	1	ABM, ASC Bootstrap Loader on fail
0	0	Generic Bootstrap Loader

5.1.5.2 Configuration by Boot Mode Index (BMI)

The Boot Mode Index (BMI) is 2 Byte value holding information about start-up mode of the device and Lockstep mode enable/disable.
The data structure of BMI is described below, followed by the definition of FLASH0_PROCOND bits which control RAM initialization, oscillator circuit (OSC) configuration and ESR0 pin handling by SSW.
***NOTE**: The start-up mode selections defined as "Reserved" in BMI.HWCFG are treated as invalid by SSW*

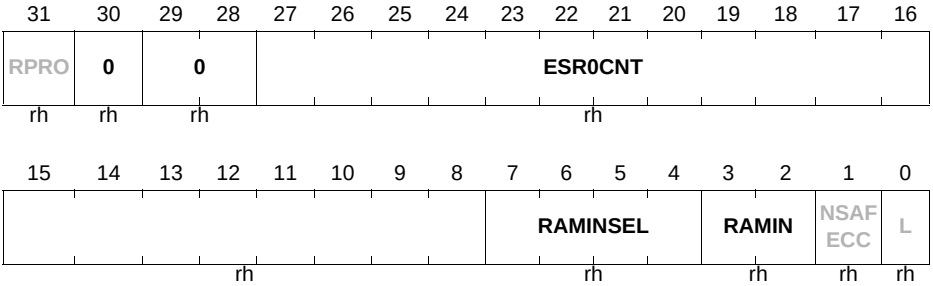
BMI structure

BMI
Boot Mode Index Boot Mode Header offset 04_H)



Field	Bits	Typ	Description
0	[15:10], 7, [2:0]	r	**Reserved**
PINDIS	3	r	**Mode selection by configuration pins:** 0 Mode selection by HWCFG pins is enabled 1 Mode selection by HWCFG pins is disabled
HWCFG	[6:4]	r	**Start-up mode selection:** 111_B Internal start from Flash 110_B Alternate Boot Mode (ABM) 101_B Reserved 100_B Generic Bootstrap Loader 011_B ASC Bootstrap Loader 010_B Reserved 001_B Reserved 000_B Reserved
LCL0LSEN	8	r	**Lockstep Comparator Logic control for CPU0:** 0 Lockstep is disabled for CPU0 1 Lockstep is enabled for CPU0
LCL1LSEN	9	r	**Reserved in TC21x/TC22x/TC23x**

FLASH0_PROCOND
DFlash Protection Configuration (F800 1030$_H$)



Field	Bits	Type	Description
NSAFECC, L, RPRO	0,1,31	rh	Flash related control bits

Field	Bits	Type	Description
RAMIN	[3:2]	rh	**RAM initialization by SSW control:** $x0_B$ Initialize RAMs as selected by RAMINSEL upon cold power-on $0x_B$ Initialize RAMs as selected by RAMINSEL upon warm power-on
RAMINSEL	[7:4]	rh	**RAMs selected for initialization by SSW:** $xxx0_B$ PSPR, DSPR and PCACHE in CPU0 are selected for initialization $xx0x_B$ Reserved in TC21x/TC22x/TC23x $x0xx_B$ Reserved in TC21x/TC22x/TC23x $0xxx_B$ LMU and DAM RAMs are selected for initialization
ESR0CNT	[27:16]	rh	**ESR0 prolongation counter** FFF_H Application Reset Indicator/ESR0 pin is not handled by SSW 000_H Application Reset Indicator - ESR0 pin when respectively configured - is released by SSW as soon as possible (zero prolongation) after device reset else Application Reset Indicator - ESR0 pin at low level, when respectively configured - is hold by SSW active for (ESR0CNT)*10µsec after device reset release
0	[30:28, 15:8]	rh	**Reserved**

5.1.6 Startup Software Main Flow

Below the SSW functionality is outlined during different execution-steps (refer to **Figure 5-3**) and upon various configuration settings.

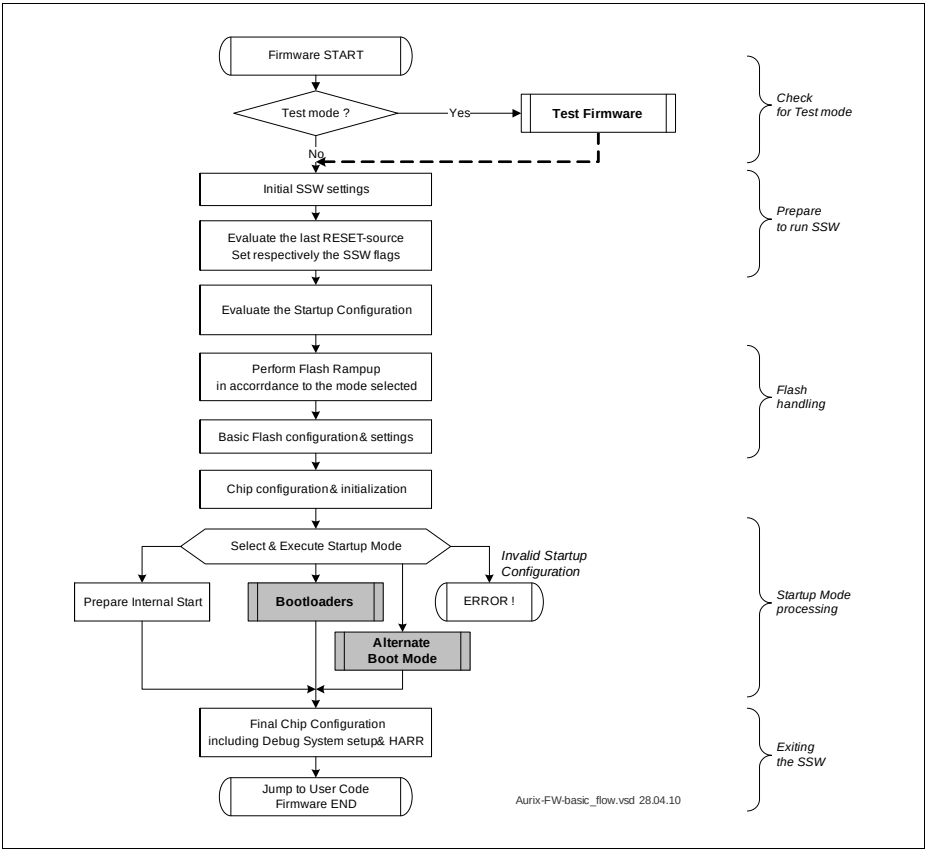


Figure 5-3 TC23x Firmware: Firmware main flow

5.1.6.1 Basic Device Settings

The target of this functional module is to initialize several TC23x-registers with the values, which will be first seen by the user - or generally available - after exiting the SSW.

5.1.6.2 RAMs Handling

Only HSM RAM is initialized after any power-on by Startup Software in TC23x.

For other RAMs initialization upon cold and/or warm power-on can be requested by **FLASH0_PROCOND** (refer to **Chapter 5.1.5.2**).

5.1.6.3 Select and Prepare Startup Modes

TC21x/TC22x/TC23x User Startup Configurations and modes are summarized below.

Internal Start

This is the basic TC23x type of operation in which the user code is started out of the Internal Flash Memory.

The User Start Address is set to A000'0020$_H$.

Bootstrap Loader Modes

The selected Bootstrap Loader (these routines are described in **Chapter 10.2**) is executed only if the SSW-flag “Reset Configuration Updated” is set. This is to avoid multiple executions of the Bootstrap Loader and to start directly the code already downloaded after some - intended to be “application only” - reset events, for example after a Watchdog Timer reset which has been configured to trigger application reset.

The supported Bootloader selections are:

- ASC Bootloader - ASC communication protocol via ASC pins
- Generic Bootloader via CAN pins - the communication protocol is automatically selected by the SSW between ASC and CAN

After downloading (in case) the code, the User Start Address STADD is set to the beginning of Program Scratchpad RAM PSPR at C000 0000$_H$.

Alternate Boot Mode

The handling of this (ABM) start-up mode is according to the flow at **Figure 5-1** and **Figure 5-2**.

If this mode is selected by a valid BMI Header (or by pins if pin configuration is enabled in a valid Header) and the code check pass - the User Start Address STADD is set to the respective value from the header (STADABMx).

If the code check fails, the handling depends how ABM was selected:

- if selected by BMI - this BMI is considered then as not valid and not used for configuration, the next header (of any) is taken for evaluation
- if selected by pins and no more Headers are available - Generic or ASC Bootloader mode is taken in accordance to the pin-selection (refer to **Chapter 5.1.5.1**)

Secure Boot option handling

If HSM module is available for the user (according to device configuration) - Secure Booting is supported in which the user code is processed by HSM.

SSW supplies information about the current device mode, reset type, the primary selected and the effectively taken start-up mode, which information is available for HSM module to perform accordingly boot and user code processing.

5.1.6.4 Final Chip Settings

The last device configuration steps performed by the SSW include:

Flash access handling

SSW flow here includes:

- fetches (code/data, from Program/Data/HSM Flash) control
  - if user code will be started from internal flash - all fetches are enabled
  - otherwise - according to the protection (active/not) state
- Program and Erase operations are enabled

No handling of Unique Chip ID and Calibration data

This information (Unique Chip ID and Calibration data) is now stored in User Configuration Block which is readable for the user. As consequence it is no more installed into DSPR but can be read directly by the application software.

Debug System handling

As first point of this processing, the SSW internal flag Unlock Debug Interface is set if debug access to device will be enabled according to the following evaluation sequence:

1. SSW checks either an external tool has requested debug access by writing a defined content (32-bit value CMD_KEY_EXCHANGE=$76D6E24A_H$ for TC21x/TC22x/TC23x) into COMDATA register
   a) if yes - continue with the next step
   b) if not - go to step 4.
2. still in Cerberus Communication mode, SSW confirms request reception and receives 8 further words through COMDATA register
3. the data received (256 bits) is sent by SSW to PMU to be checked as debug interface password and the result is evaluated by SSW:
   a) if OK - debug password is correct, go to step 6.
   b) if fail - continue with the next step
4. install into COMDATA 32-bit value serving for an external tool to identify the device connected - UNIQUE_CHIP_ID_32BIT
5. check if Flash protection is activated:

a) if yes - debug interface will be left locked, no debug access to device, exit the sequence
b) if not - set SSW internal flag, debug interface will be unlocked, exit the sequence

The following steps are introduced to support so called “Debug Self-destructive Entry” feature. The target is, if this feature is activated and debugging is requested by correct debug password (see steps 1...3 above) - to destroy vital functions of the device (so that it is not usable in a real application anymore) before effectively granting debug access. The steps performed are:

6. check either “Debug Self-destructive Entry” feature is activated for the device (PROCONHSMOTP.DESTDBG=11_B):
   a) if yes - continue with the next step
   b) if no - debug access granted, go to step 5.b)
7. using PMU command sequences, erase and re-program UCB_DBG with the previous password but setting “Entered Debug Mode” marker (EDM=11_B) in PROCONDBG; then go to step 5.b) - debug access granted

Next, Halt after Reset is prepared if requested. The SSW processing here is as follows:

- check either external (debug) access to the device will be generally granted
  – if Not -> exit this procedure
- check either Halt After Reset is requested
  – if Not -> exit this procedure
- configure a Break After Make breakpoint at the last SSW instruction
- enable On-Chip Debug Support system.

Note: The final debug-related operation - unlocking (in case) the debug interface - is performed later.

ESR0 pin handling

If both the conditions

- ESR0CNT<>FFF_H in **FLASH0_PROCOND** (refer to register description in **Chapter 5.1.5.2**) AND
- ESR0-pin configuration upon SSW entry is open-drain reset output (SCU_IOCR.PC0 in [1110_B, 1010_B])

are satisfied, SSW will:

- release ESR0 pin - by installing SCU_ESROCFG.ARC:=1 which clears Application Reset Indicator in SCU_ESROCFG.ARI - with some delay after device internal reset is released (i.e. CPU0 started), the delay is defined as follows
  – if **FLASH0_PROCOND**.ESR0CNT=000_H - about 500µsec upon cold power-on, not longer than 20µsec otherwise
  – if **FLASH0_PROCOND**.ESR0CNT=001_H...FFE_H - (ESR0CNT)*10µsec
- wait until ESR0 pin is effectively high - indicated by SCU_IN.P0=1 - at the very SSW end and before jumping to the first user instruction

To generate configurable ESR0 delay after device reset release, SSW uses System Timer 0 (STM0) and especially takes into account the following default settings after power-on/system reset:

- STM is reset and starts counting from zero
- STM is clocked with 50MHz i.e. $f_{STM}=F_{BACK}/2$

Attention: Both the above conditions could not be true after application reset, if default settings are changed by user code executed after power-on/system reset. In such a case, ESR0 handling by SSW will not work correctly, meaning the real prolongation will not correspond to ESR0CNT value configured.

Lockstep configuration

Upon cold power-on only SSW performs Lockstep configuration as follows:

- if valid BMI has been found during start-up mode evaluation:
  - Lockstep control for CPU0 is installed in LCLCON0.LSEN from BMI[8]
- if no valid BMI has been found during start-up mode evaluation:
  - Lockstep control for CPU0 is disabled by installing LCLCON0.LSEN=0

5.1.6.5 Ending the SSW and Starting the User Code

The last steps executed by the SSW are:

- activate the Startup Protection
- unlock (in case) Debug Interface
- jump to the first User Instruction at address STADD.

Additionally, if all the following conditions are satisfied:

- the device is an ED
- debug access to device is allowed
- halt after reset is not requested
- the last reset has been a power-on (cold or warm)

SSW performs at its very end additional checks and in case all these succeed - the last SSW instruction jumps not to the “standard” STADD but to an address inside EMEM (Emulation Memory). This SSW part implements the sequence defined in TC21x/TC22x/TC23x ED Target Specification, Chapter “Startup with prolog code in EMEM”.

5.2 Bootstrap Loaders

These routines provide mechanisms to load an user program via selected interface by moving code into PMI Scratchpad RAM (an Internal Program memory). The loaded code is started after exiting the BootROM.

5.2.1 ASC Bootstrap loader

The ASC Bootloading routine implements the following steps:

- RxD/TxD pins configuration is done in accordance to the TC23x definitions, as well as depending either the routine is invoked upon “ASC Bootloader”-startup mode (ASC-only pins are used) or following an ASC-protocol detection upon “Generic Bootloader”-mode (CAN/ASC-shared pins are used but configured to ASC module)
- baudrate calculation is done based on the zero Byte sent by the host
- ASCLIN channel n (refer to **Table 5-4**) is initialized (without enabling the receiver) to the baudrate as determined, 8 data and 1 stop bit
- acknowledge byte $D5_H$ is sent to the host indicating the device is ready to accept a data transfer
- after the acknowledge byte is transmitted, the receiver is enabled
- the bootloader enters a loop waiting to receive exactly 128 bytes which are stored as 32 words in CPU0 Program Scratchpad RAM starting from address C000 0000_H

Once 128 bytes are received, the SSW continues further - refer to **Figure 5-3**. After exiting the SSW, user code will be started from address C000 0000_H (CPU0_PSPR).

5.2.2 CAN Bootstrap Loader

The CAN bootstrap loader transfers program code/data via node 1 of the MultiCAN module into CPU0 Program Scratchpad RAM.

The CAN Bootstrap Loader transfers data from an external host to the TC21x/TC22x/TC23x using eight-byte data frames. The number of data frames to be received is programmable and determined by the 16-bit data message count value DMSGC.

Attention: When selected upon power-on or system reset, CAN Bootstrap Loader requires that an external clock source/oscillator is connected to XTAL pins of the device.

The communication between TC21x/TC22x/TC23x and external host is based on the following three CAN standard frames:

- Initialization frame - sent by the external host to the TC21x/TC22x/TC23x
- Acknowledge frame - sent by the TC21x/TC22x/TC23x to the external host
- Data frame(s) - sent by the external host to the TC21x/TC22x/TC23x

The initialization frame is used in the TC21x/TC22x/TC23x for baud rate detection. After a successful baud rate detection is reported to the external host by sending the acknowledge frame, data is transmitted using data frames.

Initialization Phase

The first task is to determine the CAN baud rate at which the external host is communicating. This task requires the external host to send initialization frames continuously to the TC21x/TC22x/TC23x. The first two data bytes of the initialization frame include a 2-byte baud rate detection pattern (5555_H), an 11-bit (2-byte) identifier ACKID for the acknowledge frame, a 16-bit data message count value DMSGC, and an 11-bit (2-byte) identifier DMSGID to be used by the data frame(s).

The CAN baud rate is determined by analyzing the received baud rate detection pattern (5555_H) and the baud rate registers of the MultiCAN module are set accordingly. The TC21x/TC22x/TC23x is now ready to receive CAN frames with the baud rate of the external host.

Acknowledge Phase

In the acknowledge phase, the bootstrap loader waits until it receives the next correctly recognized initialization frame from the external host, and acknowledges this frame by generating a dominant bit in its ACK slot. Afterwards, the bootstrap loader transmits an acknowledge frame back to the external host, indicating that it is now ready to receive data frames. The acknowledge frame uses the message identifier ACKID that has been received with the initialization frame.

Data Transmission Phase

In the data transmission phase, data frames are sent by the external host and received by the TC21x/TC22x/TC23x. The data frames use the 11-bit data message identifier DMSGID that has been sent with the initialization frame. Eight data bytes are transmitted with each data frame. The first data byte is stored in Program Scratchpad RAM starting from address $C000\ 0000_H$. Consecutive data bytes are stored at incrementing addresses.

Both communication partners evaluate the data message count DMSGC until the requested number of CAN data frames has been transmitted.

After the reception of the last CAN data frame, the SSW continues further - refer to **Figure 5-1**. After exiting the SSW, user code will be started from address $C000\ 0000_H$ (CPU0_PSPR).

5.2.3 Summary of Bootstrap Loader Modes

This table summarizes the external hardware provisions for bootstrap loader modes in TC21x/TC22x/TC23x.

Table 5-4 Configuration Data for Bootstrap Loader Modes

Bootstrap Loader Mode	Channel/ node	RxD Line	TxD Line	Transferred Data	Supported baudrates
ASC Bootstrap Loader mode	ASCLIN0	P15.3	P15.2	128 Bytes	28k...2500k
Generic Bootstrap Loader mode - ASC protocol	ASCLIN0	P14.1	P14.0	128 Bytes	28k...2500k
Generic Bootstrap Loader mode - CAN protocol	CAN1	P14.1	P14.0	8×n Bytes 1)	up to 1000k 2)

1) n = DMSGC, Data Message Count sent by the host with Initialization frame.

2) with 20MHz XTAL clock source

5.3 Shutdown request handler

All the active CPUs in TC21x/TC22x/TC23x jump unconditionally to the entry point of this handler upon any warm reset request - refer to Reset Control Unit Section in SCU ITS.

It is guaranteed by hardware, this handler can not be interrupted by any other (interrupt/trap) request, and upon its end all the CPUs are in stable passive state reached by a controlled ramp-down sequence, preventing big current jumps.

At its entry point - common for all the CPUs - the firmware causes any running CPU to jump further to its own handler. For this purpose CORE_ID register is read and because this register value is individual for any CPU - upon CORE_ID=0, 1 or 2 the firmware jumps to the routine for respective CPU.

The functionality of all handlers is similar, namely:

- prepare work data for execution of average-power loop
- execute average-power loop until SCU_RSTCON2.TOUTyy=1
- execute WAIT instruction

Upon completion of the above sequence, CPUx has reached passive state yy microseconds after shutdown request activation, whereas:

- yy=60 for CPU0

5.4 Preparation before to enter Stand-by mode

During stand-by mode preparation, the user software must do the following:

- read sequentially 16 words from the “reserved area” in CPU0_DSPR starting at address $D000'2000_H$

- for any word check either it equals FFFF FFFF$_H$ or zero
  – if yes - skip it and go to the next reserved location
  – if no
    - use this word as 32-bit address, read the data from that address and store this data back into the same reserved location
    - go to the next reserved location

6 CPU Subsystem

This chapter describes the implementation-specific options of the TriCore CPUs found in the AURIX series of devices. Details of both TriCore1.6P (TC1.6P) and TriCore1.6E (TC1.6E) CPUs are included. The CPU and local memory configurations of the various members of the AURIX family are detailed in the following table.

This chapter should be read in conjunction with the TriCore Architecture Manual. Topics covered by the architecture manual include:-

- Architectural Overview
- Programing Model
- CPU Registers
- Tasks and Functions
- Interrupt Handling
- Traps
- Memory Protection System
- Temporal Protection System
- Floating Point Operations
- Debug
- Instruction Set

6.1 AURIX Family CPU configurations

The different CPU and local memory configurations for the AURIX family of devices are detail in the following tables:-

Table 6-1 Processor and local memory configuration of the TC29x

Processor	Core -ID	Program Cache	Program Scratch Pad	Data Cache	Data Scratch Pad	Standby SRAM	Lock-Step
TC1.6P	2	32 KB	32 KB	8 KB	240 KB	No	No
TC1.6P	1	32 KB	32 KB	8 KB	240 KB	No	Yes
TC1.6P	0	16 KB	32 KB	8 KB	120 KB	Yes	No

Table 6-2 Processor and local memory configuration of the TC27x

Processor	Core -ID	Program Cache	Program Scratch Pad	Data Cache	Data Scratch Pad	Standby SRAM	Lock-Step
TC1.6P	2	16 KB	32 KB	8 KB	120 KB	No	No
TC1.6P	1	16 KB	32 KB	8 KB	120 KB	No	Yes
TC1.6E	0	8 KB	24 KB	0[1)]	112 KB	Yes	Yes

1) TC1.6E has a 128Byte read buffer in place of a data cache

Table 6-3 Processor and local memory configuration of the TC26x

Processor	Core -ID	Program Cache	Program Scratch Pad	Data Cache	Data Scratch Pad	Standby SRAM	Lock-Step
TC1.6P	1	16 KB	32 KB	8 KB	120 KB	No	Yes
TC1.6E	0	8 KB	16 KB	0[1)]	72 KB	Yes	No

Table 6-4 Processor and local memory configuration of the TC24x

Processor	Core -ID	Program Cache	Program Scratch Pad	Data Cache	Data Scratch Pad	Standby SRAM	Lock-Step
TC1.6E	0	8 KB	16 KB	0[1)]	80 KB	Yes	No

Table 6-5 Processor and local memory configuration of the TC23x

Processor	Core-ID	Program Cache	Program Scratch Pad	Data Cache	Data Scratch Pad	Standby SRAM	Lock-Step
TC1.6E	0	8 KB	8 KB	0[1)]	184 KB	Yes	Yes

Table 6-6 Processor and local memory configuration of the TC22x

Processor	Core-ID	Program Cache	Program Scratch Pad	Data Cache	Data Scratch Pad	Standby SRAM	Lock-Step
TC1.6E	0	8 KB	8 KB	0[1)]	88 KB	Yes	Yes

1) TC1.6E has a 128Byte read buffer in place of a data cache

6.2 Central Processing Unit Features

Key CPU Features include:-

Architecture

- 32-bit load store architecture
- 4 Gbyte address range (2^{32})
- 16-bit and 32-bit instructions for reduced code size
- Data types:
  - Boolean, integer with saturation, bit array, signed fraction, character, double-word integers, signed integer, unsigned integer, IEEE-754 single-precision floating point
- Data formats:
  - Bit, byte (8-bits), half-word (16-bits), word (32-bits), double-word (64-bits)
- Byte and bit addressing
- Little-endian byte ordering for data, memory and CPU registers
- Multiply and Accumulate (MAC) instructions: Dual 16 × 16, 16 × 32, 32 × 32
- Saturation integer arithmetic
- Packed data
- Addressing modes:
  - Absolute, circular, bit reverse, long + short, base + offset with pre- and post-update
- Instruction types:
  - Arithmetic, address arithmetic, comparison, address comparison, logical, MAC, shift, coprocessor, bit logical, branch, bit field, load/store, packed data, system
- General Purpose Register Set (GPRS):
  - Sixteen 32-bit data registers
  - Sixteen 32-bit address registers
  - Three 32-bit status and program counter registers (PSW, PC, PCXI)
- Debug support (OCDS):
  - Level 1, supported in conjunction with the CPS block
  - Level 3, supported in conjunction with the MCDS block (Emulation Device only).
- Flexible memory protection system providing multiple protection sets with multiple protection ranges per set.
- Temporal protection system allowing time bounded real time operation.

TC1.6P Implementation

- Most instructions executed in 1 cycle
- Branch instructions in 1, 2 or 3 cycles (using dynamic branch prediction)
- Wide memory interface for fast context switch
- Automatic context save-on-entry and restore-on-exit for: subroutine, interrupt, trap
- Four memory protection register sets
- Dual instruction issuing (in parallel into Integer Pipeline and Load/Store Pipeline)
- Third pipeline for loop instruction only (zero overhead loop)

- Single precision Floating Point Unit (IEEE-754 Compatible)
- Dedicated Integer divide unit
- Implementation optimised for performance.
- 16 data protection ranges, 8 code protection ranges

TC1.6E Implementation

- Most instructions executed in 1 cycle
- Branch instructions in 1 or 2 cycles (using static branch prediction)
- Wide memory interface for fast context switch
- Automatic context save-on-entry and restore-on-exit for: subroutine, interrupt, trap
- Four memory protection register sets
- Single instruction issue per cycle
- Single precision Floating Point Unit (IEEE-754 Compatible)
- Dedicated Integer divide unit
- Implementation optimised for power
- 16 data protection ranges, 8 code protection ranges

6.3 TC1.6P Implementation Overview

The following sections give an overview of the TC1.6P Implementation

6.3.1 CPU Diagram

The Central Processing Unit (CPU) comprises of an Instruction Fetch Unit, an Execution Unit, a General Purpose Register File (GPR), a CPU Slave interface (CPS), and Floating Point Unit (FPU).

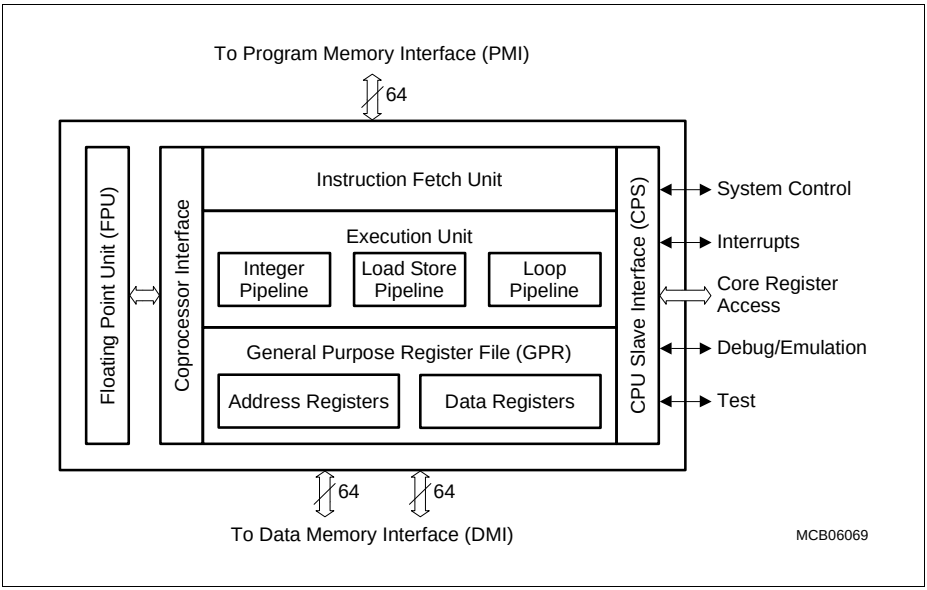


Figure 6-1 CPU Block Diagram

6.3.2 Instruction Fetch Unit

The Instruction Fetch Unit pre-fetches and aligns incoming instructions from the 64-bit wide Program Memory Interface (PMI). Instructions are placed in predicted program order in the Issue fifo. The Issue fifo buffers up to six instructions and directs the instruction to the appropriate execution pipeline.

The Instruction Protection Unit checks the validity of accesses to the PMI and the integrity of incoming instructions fetched from the PMI.

The branch unit examines the fetched instructions for branch conditions and predicts the most likely execution path based on previous branch behavior. The Program Counter Unit (PC) is responsible for updating the program counters.

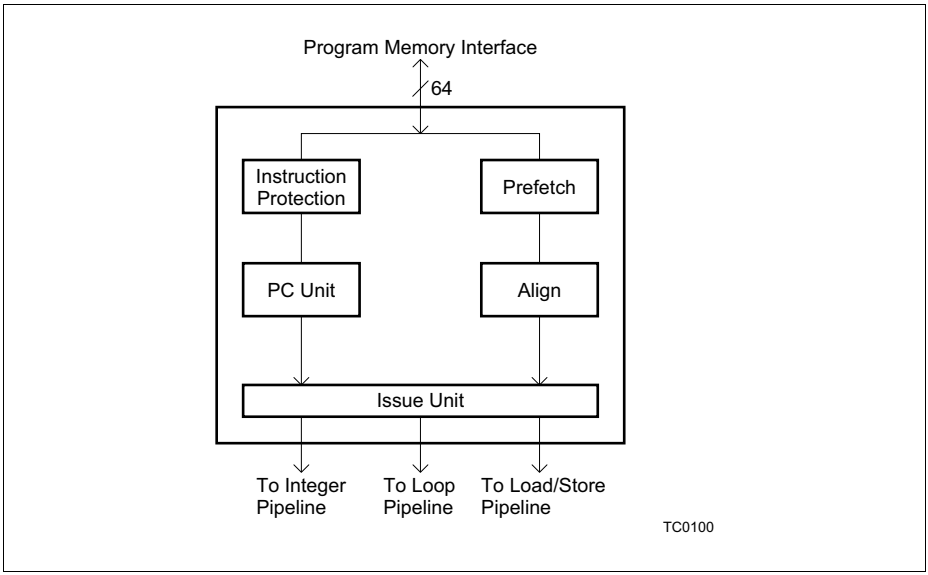


Figure 6-2 Instruction Fetch Unit

6.3.3 Execution Unit

The Execution Unit contains the Integer Pipeline, the Load/Store Pipeline and the Loop Pipeline. All three pipelines operate in parallel, permitting up to three instructions to be executed in one clock cycle. In the execution unit all instructions pass through a decode stage followed by two execute stages. Pipeline hazards (stalls) are minimised by the use of forwarding paths between pipeline stages allowing the results of one instruction to be used by a following instruction as soon as the result becomes available.

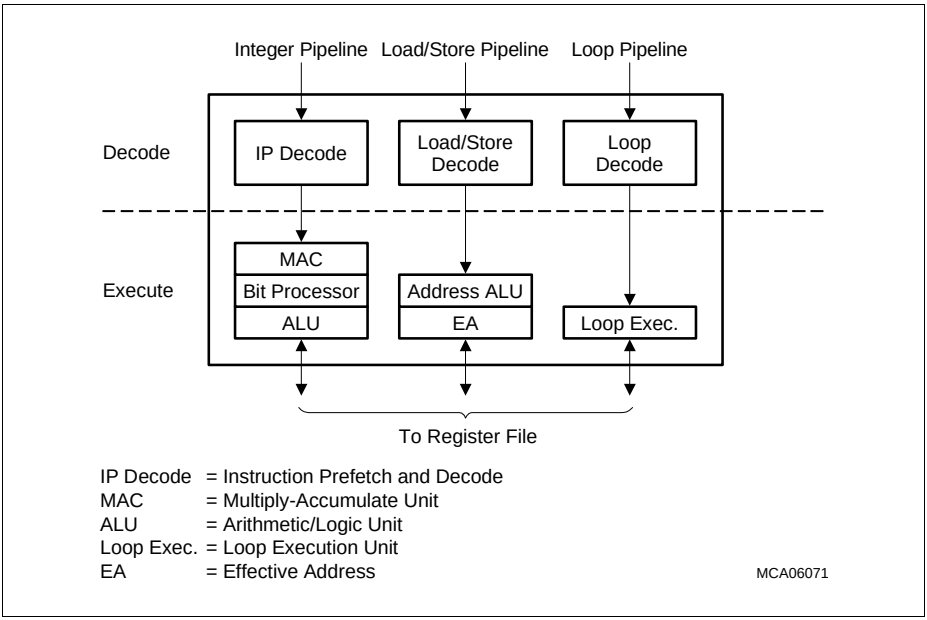


Figure 6-3 Execution Unit

6.3.4 General Purpose Register File

The CPU has a General Purpose Register (GPR) file, divided into an Address Register File (registers A0 through A15) and a Data Register File (registers D0 through D15).

The data flow for instructions issued to the Load/Store Pipeline is steered through the Address Register File.

The data flow for instructions issued to/from the Integer Pipeline and for data load/store instructions issued to the Load/Store Pipeline is steered through the Data Register File.

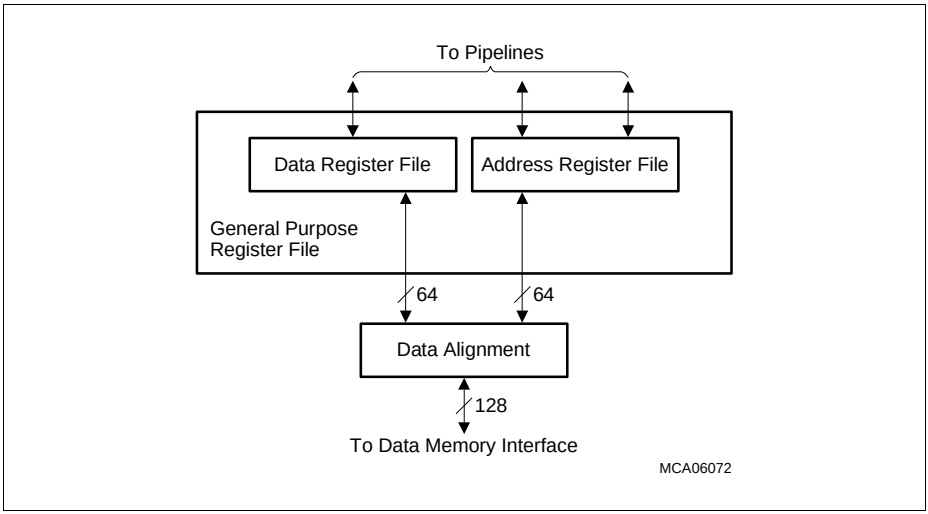


Figure 6-4 General Purpose Register File

6.4 TC1.6E Implementation Overview

The following sections give an overview of the TC1.6E Implementation

6.4.1 CPU Diagram

The Central Processing Unit (CPU) comprises of an Instruction Fetch Unit, an Execution Unit, a General Purpose Register File (GPR), a CPU Slave interface (CPS), and Floating Point Unit (FPU).

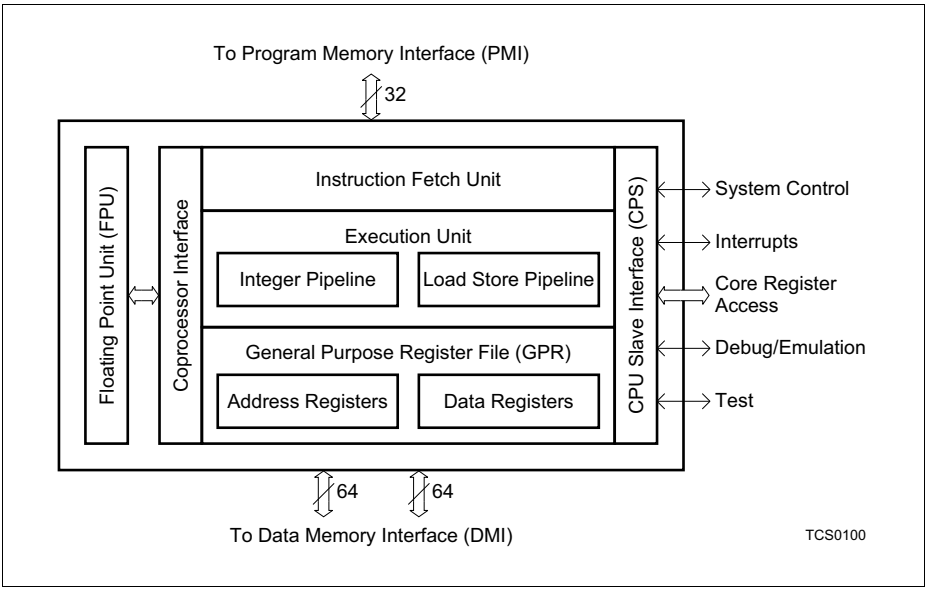


Figure 6-5 CPU Block Diagram

6.4.2 Instruction Fetch Unit

The Instruction Fetch Unit pre-fetches and aligns incoming instruction half-words from the 32-bit wide Program Memory Interface (PMI). The Issue Unit directs a single aligned instruction to the appropriate execution pipeline.

The Instruction Protection Unit checks the validity of accesses to the PMI and the integrity of incoming instructions fetched from the PMI.

The Program Counter Unit (PC) is responsible for updating the program counters.

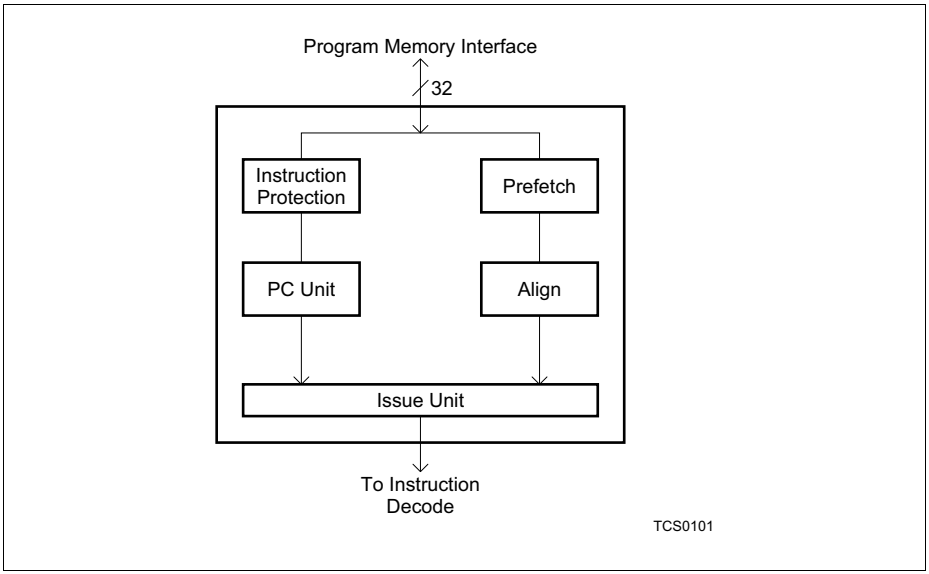


Figure 6-6 Instruction Fetch Unit

6.4.3 Execution Unit

The Execution Unit contains the Integer Pipeline and the Load/Store Pipeline. In TC1.6E, loop instructions are always executed by the Load/Store pipeline.

The TC1.6E issues a single instruction per clock cycle, and as such no more than one instruction will be executed in one clock cycle.

In the execution unit all instructions pass through a decode stage followed by two execute stages. Pipeline hazards (stalls) are minimised by the use of forwarding paths between pipeline stages allowing the results of one instruction to be used by a following instruction as soon as the result becomes available.

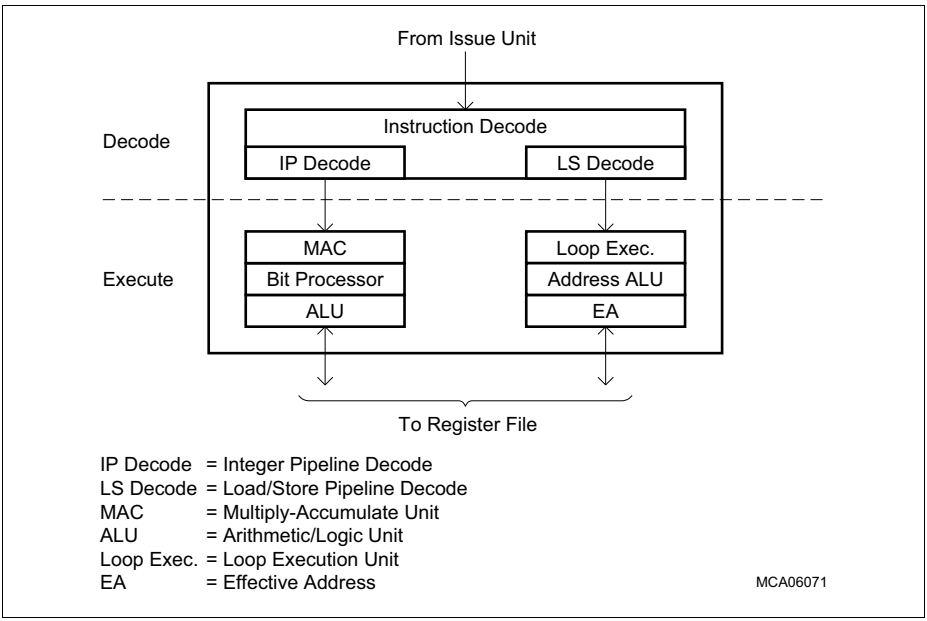


Figure 6-7 Execution Unit

6.4.4 General Purpose Register File

The CPU has a General Purpose Register (GPR) file, divided into an Address Register File (registers A0 through A15) and a Data Register File (registers D0 through D15).

The data flow for instructions issued to the Load/Store Pipeline is steered through the Address Register File.

The data flow for instructions issued to/from the Integer Pipeline and for data load/store instructions issued to the Load/Store Pipeline is steered through the Data Register File.

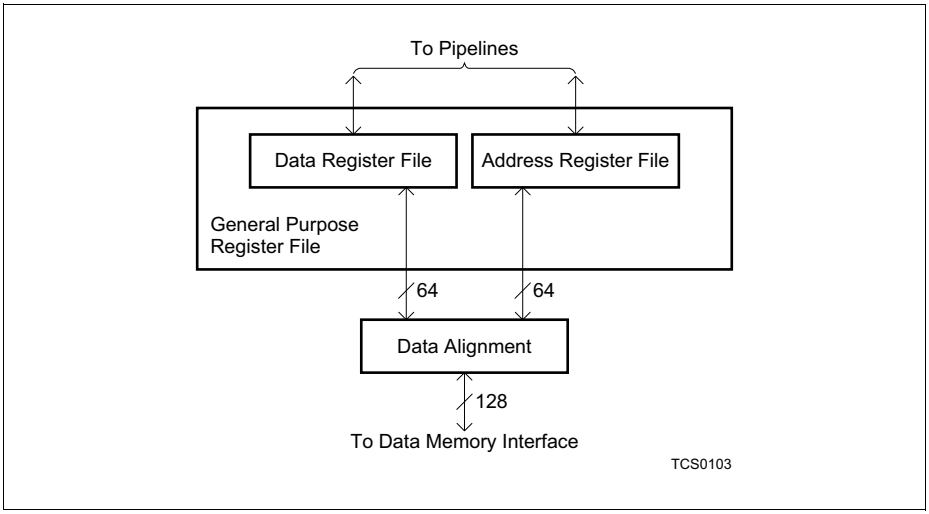


Figure 6-8 General Purpose Register File

6.5 Summary of functional changes from TC1.3.1

The TC1.6P and TC1.6E CPUs utilise different pipeline organisations than that used in the TC1.3. One effect of the new pipeline organisation is to increase the load-use penalty to 1 from 0. This necessitates re-scheduling of code to achieve optimum performance.

Other significant adaptations to the existing TC1.3.1 CPU are as follows:

- Fully Pipelined Floating Point Unit (FPU)
  - Most floating point instructions now have a repeat rate of 1
- Improved debug system - now decoupled from protection system.
  - 8 comparators proving up to 4 ranges, selectable for PC or load-store address
- Expanded and enhanced memory protection unit (MPU)
  - 16 data ranges and 8 code ranges.
- New Temporal protection system.
  - Guards against task runtime overrun.
- New Safety protection system. Tasks identified as safe by new PSW bit (PSW.S)
- New instructions for improved Interrupt and Data Cache manipulation support.
  - DISABLE, RESTORE, CACHEI.I
- New instructions for Fast Integer Divide
  - DIV, DIV.U
- New Instructions for fast call and return with minimal saving of state.
  - FCALL,FCALLA,FCALLI, FRET
- Long offset addressing mode introduced for byte, half word and address accesses.
  - LD.BU, LD.B, LD.HU, LD.H, ST.B, ST.H, ST.A
- Extended range of 16 bit jumps
  - JEQ, JNE
- New Synchronisation Instructions
  - CMPSWAP.W, SWAPMSK.W
- New CRC instruction
  - CRC32
- New wait for interrupt instruction
  - WAIT
- Increased flexibility in the system address map.
- Full SECDED ECC protection for all scratch, cache and tag memory structures.
- Cache and Scratchpad memory systems now entirely separated.
  - Cache memories may be mapped as additional scratchpad.
- Selectable interrupt vector table size (32bytes/entry, 8bytes/entry).

6.6 CPU Implementation-Specific Features

This section describes the implementation-specific features of the TC1.6P and TC1.6E CPUs. For a complete description of all registers, refer to the TriCore Architecture Manual.

6.6.1 Context Save Areas / Context Operations

The CPU uses a uniform context-switching method for function calls, interrupts and traps. In all cases the Upper Context of the task is automatically saved and restored by hardware. Saving and restoring of the Lower Context may be optionally performed by software.

In TC1.6P, Context Save Areas (CSA) and addresses targeted by explicit context load/store instructions (e.g. LDLCX) may be placed in DSPR or external memory (cached or uncached).

In TC1.6E, Context Save Areas (CSA) and addresses targeted by explicit context load/store instructions may be placed in DSPR only.

CSA Placement in DSPR

The actual timing of context operations is dependent upon the placement of the Context Save Areas. Maximum performance is achieved when the Context Save Area is placed in DSPR. In this case all context save and restores operations take four cycles.

CSA Placement in Cached External Memory (TC1.6P Only)

In this case, the timing is also dependent on the state of the Data Cache. The best case Data Cache operation occurs when context saves do not incur a cache line writeback, and context restores hit in the data cache. In this case all context saves and restores take eight cycles.

6.6.2 Program Counter (PC) Register

The Program Counter (PC) holds the address of the instruction that is currently fetched and forwarded to the CPU pipelines. The CPU handles updates of the PC automatically.

Software can use the current value of the PC for various tasks, such as performing code address calculations. Reading the PC through software executed by the CPU must only be done with an MFCR instruction. Such a read will return the PC of the MFCR instruction itself. Explicit writes to the PC through an MTCR instruction must not be done due to possible unexpected behavior of the CPU. The PC may be written only when the CPU is halted.

The CPU must not perform Load/Store instructions to the mapped address of the PC in Segment 15. A MEM trap will be generated in such a case. Bit 0 of the PC register is read-only and hard-wired to 0.

6.6.3 Store Buffers

To increase performance the TC1.6P/E CPUs implements store buffering to decouple memory write operations from CPU instruction execution. All stores from the CPU are placed in the store buffer prior to being written to local memory or transferred via the bus system. Write data is taken from the store buffers and written to memory when the target memory or bus interface becomes available. In normal operation the CPU will prioritise memory load operations over store operations in order to improve performance unless:-

- The store buffer is full.
- The load is to peripheral space and a store to peripheral space exists in the store buffer. (In order peripheral space access).
- The load or store is part of an atomic operation.

Typically the operation of the store buffer is invisible to the end user. If there is a requirement that data is written to memory prior to execution of a subsequent instruction then a DSYNC instruction may be used to flush the store buffers.

To further improve performance consecutive byte writes to the same half word location are merged in the store buffer.

The TC1.6P CPU store buffer can hold the data for up to 6 stores. The TC1.6E CPU store buffer can hold the data for up to 2 stores.

Store buffer operation may be disabled be setting the SMACON.IODT bit. This should not be done in normal execution as it will severely limit performance.

6.6.4 Interrupt System

An interrupt request can be generated by the on-chip peripheral units, or it can be generated by external events. Requests can be targeted to any CPU.

The interrupt system evaluates service requests for priority and to identify whether the CPU should receive the request. The highest-priority service request is then presented to the CPU by way of an interrupt.

On taking an interrupt the CPU will vector to a unique PC generated from the interrupt priority number and the Base Interrupt Vector (BIV). The spacing between the vector PCs in the interrupt vector table may be selected to be either 32 Bytes or 8 Bytes using BIV[0].

Both the TC1.6P and TC1.6E implement a fast interrupt system. This system avoids unnecessary context save and restore operations and hence speeds up interrupt routine entry. A fast interrupt is triggered when:-

- An Interrupt is pending
- A Return From Interrupt instruction is being executed (RFE)
- The priority of the pending interrupt is greater than the priority level that would be returned to should the RFE be executed (ICR.PIPN > PCXI.PCPN).
- Interrupts will be enabled should the RFE be executed. (PCXI.PIE == 1)
- Fast Interrupts are not otherwise disabled due to the presence of MTCR instructions or context operations in the pipeline.

Without the fast interrupt operation the RFE would cause a restore of the upper context immediately followed by a save of the same upper context back to exactly the same memory location (Minimum 8 cycles). The fast interrupt system replaces this redundant restore/save sequence with a load of PCXI/PSW/A10/A11 (Minimum 1 Cycle) from the saved context. System state at the end of the fast interrupt is the same as if the standard RFE/Interrupt sequence had been performed.

6.6.5 Trap System

The following traps have implementation-specific properties.

UOPC - Unimplemented Opcode (TIN 2)

The UOPC trap is raised on optional MMU instructions, coprocessor two and coprocessor three instructions.

OPD - Invalid Operand (TIN 3)

The CPU raised OPD traps for instructions that take even-odd register pairs as an operand where if the operand specifier is odd.

DSE - Data Access Synchronous Error (TIN 2)

The Data Access Synchronous Bus Error (DSE) trap is generated by the DMI module when a load access from the CPU encounters certain error conditions, such as a Bus error, or an out-of-range access to DSPR. When a DSE trap is generated, the exact cause of the error can be determined by reading the Data Synchronous Trap Register, DSTR. For details of possible error conditions and the corresponding flag bits in DSTR, see **“DMI Trap Generation” on Page 6-103**.

DAE - Data Access Asynchronous Error (TIN 3)

The Data Access Asynchronous Error Trap (DAE) is generated by the DMI module when a store or cache management access from the CPU encounters certain error conditions, such as an Bus error. When a DAE trap is generated, the exact cause of the error can be determined by reading the Data Asynchronous Trap Register, DATR. For details of possible error conditions and the corresponding flag bits in DATR, see **“DMI Trap Generation” on Page 6-103**.

PIE Program Memory Integrity Error (TIN 5)

The PIE trap is raised whenever an uncorrectable memory integrity error is detected in an instruction fetch from a local memory or the SRI bus. The trap is synchronous to the erroneous instruction. The trap is of Class-4 and has a TIN of 5.

Program memories are protected from memory integrity errors on a 64 bit basis (TC1.6P) or 32 bit basis (TC1.6E). A PIE trap is raised when an attempt is made to execute an instruction from any fetch group containing a memory integrity error.

The PIEAR and PIETR registers may be interrogated to determine the source of any error more precisely.

DIE Data Memory Integrity Error (TIN 6)

The DIE trap is raised whenever an uncorrectable memory integrity error is detected in a data access to a local memory or the SRI bus. The trap is of Class-4 and has a TIN of 6.

DIE traps are always asynchronous independent of the operation which encountered the error.

A DIE trap is raised if any memory half word (local memory) or double word (SRI bus) accessed by a load/store operation contains an uncorrectable error. The DIEAR and DIETR registers may be interrogated to determine the source of any error more precisely.

MPX Memory Protection Execute (TIN 4)

The MPX trap is raised whenever a program attempts to execute an instruction from a memory area for which it does not have execute permission and memory protection is enabled. The trap is of Class-1 and has a TIN of 4.

The TC1.6E compares the PC of the instruction with the range(s) defined by the memory protection system to determine whether or not execution is permitted.

The TC1.6P compares the 64bit aligned fetch group address with the range(s) defined by the memory protection system to determine whether or not execution is permitted.

6.6.6 Memory Integrity Error Handling

The TriCore CPUs contain integrated support for the detection and handling of memory integrity errors. The handling of memory integrity errors for the various memory types in the CPU is as follows:

6.6.6.1 Program Side Memories

The program side memories of the CPU consist of two independent memory structures:- The Program Scratchpad RAM (PSPR) and Program Cache (PCACHE). Both memory structures are ECC protected from memory integrity errors on a 64bit basis (TC1.6P) or on a 32 bit basis (TC1.6E). Any sub-width write access to the PSPR from the Bus interface is converted to a Read-Modify-Write sequence by the PMI module.

Program Scratchpad RAM (PSPR)

The Scratchpad RAM of the CPU is protected from memory integrity errors on a 64bit basis (TC1.6P) or 32 bit basis (TC1.6E). ECC protection of is enabled via the MBIST ECCS register.

For instruction fetch requests from the TriCore CPU to PSPR, the ECC bits are read along with the data bits and are passed to the CPU along with their corresponding instructions. Whenever an attempt is made to issue an instruction containing an uncorrectable memory integrity error a synchronous PIE trap is raised. The trap handler is then responsible for correcting the memory entry and re-starting program execution.

For PSPR read operations from the Bus interface, either from the DMI module or another Bus master agent, an access that results in the detection of an uncorrectable memory integrity error in the requested data causes a bus error to be returned for the bus transaction. Since the TriCore CPU may not be involved in the transaction, a separate error is also flagged to the SCU module to optionally generate an NMI trap back to the CPU.

Writes to program scratchpad memory are only ever performed from the bus interface. For write operations less than the protection width of the PSPR the memory transaction is transformed into a read-modify-write sequence inside the PMI module. Such a write

operation may result in the detection of an uncorrectable memory integrity errors during the read phase which are handled as standard read operations.

Program Cache (PCACHE)

The Scratchpad RAM of the CPU is protected from memory integrity errors on a 64bit basis (TC1.6P) or 32 bits (TC1.6E).ECC protection of is enabled via the MBIST ECCS register.

For instruction fetch requests from the TriCore CPU to PCACHE, the ECC bits are read along with the data bits of all cache ways, and an uncorrectable error signal generated for each cache way. In the case of a tag hit, the uncorrectable error signals for the corresponding cache way are passed to the CPU along with their corresponding instructions. Whenever an attempt is made to issue an instruction containing an uncorrectable error a synchronous PIE trap is raised. The trap handler is then responsible for checking the source of the memory integrity error.

Program Tag (PTag)

ECC protection of is enabled via the MBIST ECCS register.

For instruction fetch requests from the TriCore CPU to PCACHE, the program tag ECC bits are read along with the data bits and an error flag is computed. A way hit is triggered only if the tag address comparison succeeds, the valid bit is set and no ECC error in the associated tag way is detected, any other result is considered a miss. In the normal case where no error is detected in either cache way then the cache line is filled/refilled as normal. In the case where an error is detected the cache controller replacement algorithm forces the way indicating an error to be replaced. In the case where one cache way flags a cache hit, and another cache way detects an uncorrectable ECC error, the error condition is masked and has no effect on the memory integrity error handling mechanisms.

6.6.6.2 Data Side Memories

The TC1.6P CPU implements both data scratch and data cache memories. The TC1.6E implements a data scratch memory but replaces the data cache with a data read buffer (DRB). All data memory structures are ECC protected from memory integrity errors on a per-half word basis. Any byte write access to either DSPR or DCache (TC1.6P) is converted to a halfword Read-Modify-Write sequence. In normal operation isolated byte write transactions to the data memories result in no additional stall cycles.

Data Scratchpad Ram (DSPR)

The DSPR memory of both TC1.6P and TC1.6E are protected from memory integrity errors on a per-halfword basis. ECC protection of is enabled via the MBIST ECCS register.

For data load requests from the TriCore CPU to DSPR, the ECC bits are read along with the data bits and an uncorrectable error signal is generated for each half-word. If an error is detected associated with any of the data half-words passed to the CPU an error is flagged to the CPU. If such an error condition is detected an asynchronous DIE trap is raised. The trap handler is then responsible for correcting the memory entry, or for taking alternative action (such as system soft reset) if correction of the data is not possible.

For DSPR read operations from the Bus interface, either from the PMI module or another Bus master agent, an access that results in the detection of an uncorrectable error in the requested data half-words causes a bus error to be returned for the bus transaction. Since the TriCore CPU may not be involved in the transaction, a separate error is also flagged to the SCU module to optionally generate an NMI trap back to the CPU.

For write operations to DSPR of half-word size or greater, the ECC bits are pre-calculated and written to the memory in parallel with the data bits. For byte write operations the memory transaction is transformed into a half-word read-modify-write sequence inside the DMI module. As such, byte write operations may result in the detection of uncorrectable memory integrity errors, which are handled as per standard read operations.

Data Cache (DCache) - TC1.6P Only

ECC protection of is enabled via the MBIST ECCS register.

For data load requests from the TriCore CPU to DCache, the ECC bits are read along with the data bits of both cache ways, and an uncorrectable error flag computed for each half-word of each cache way. In the case where an error is detected with any of the requested data half-words in a cache way which has a corresponding tag hit, an error is flagged to the CPU. If such an error condition is detected an asynchronous DIE trap is raised. The trap handler is then responsible for correcting the memory entry, or for taking alternative action (such as system soft reset) if correction of the data is not possible.

For write operations of half-word size or greater, the check bits are pre-calculated and written to the memory in parallel with the data bits. For byte write operations the memory transaction is transformed into a half-word read-modify-write sequence inside the DMI module. As such, byte write operations may result in the detection of uncorrectable memory integrity errors as for read operations.

For cache line writeback, uncorrectable error detection is performed as dirty data is transferred to the store buffers. In all cases (normal cache line eviction, cachex.xx instruction) where an error condition is detected in a valid cache line a DIE trap is raised. The trap handler is then responsible for taking corrective action (such as system soft reset) since correction of the data is not possible.

Data Tag (DTag) - TC1.6P Only

ECC protection of is enabled via the MBIST ECCS register.

For data load or store requests from the TriCore CPU to DCache, the data tag ECC bits are read along with the data bits and an uncorrectable error flag is computed. A way hit is triggered only if the tag address comparison succeeds, the tag location is valid and no uncorrectable error in the associated tag way is detected, any other result is considered a miss. In the normal case where no error is detected in either tag way then the cache line is filled/refilled as normal. In the case of a cache miss where an error is detected in one of the tag ways and the cache line does not contain dirty data the cache controller replacement algorithm forces the way indicating an error to be replaced when the refill operation returns. In the case where one cache way flags a cache hit, and the another way detects an uncorrectable error, the error condition is masked and has no effect on the memory integrity error handling mechanisms. If a cache miss occurs, with an uncorrectable error detected on the associated data tag way and dirty data detected, then an asynchronous DIE trap is signalled to the CPU and any writeback / refill sequence aborted. The trap handler is responsible for invalidating the cache line and processing any associated dirty data if possible, or taking other corrective action. Similar action is taken for forced cache writeback using the cache manipulation instructions.

6.6.6.3 Memory Initialisation

To avoid the generation of spurious ECC errors the DSPR, PSPR must be fully initialised prior to use. This may be done either by software or by automatically by hardware (see the PMU.PROCOND register for details). The entire physical memory as detailed in **Chapter 6.1** must be initialised irrespective of any memory size variants produced for derivative products.

6.6.7 WAIT Instruction

The WAIT instruction will suspend execution until the occurrence of one of the following events.

- Enabled Interrupt
- Non-Maskable Interrupt
- Asynchronous Trap
- Idle Request
- Suspend Request
- Asynchronous Debug Halt or Trap Request

6.6.8 Instruction Memory Range Limitations

To ensure the processor cores are provided with a constant stream of instructions the Instruction Fetch Units will speculatively fetch instructions from up to 64 bytes ahead of the current PC.

If the current PC is within 64 bytes of the top of an instruction memory the Instruction Fetch Unit may attempt to speculatively fetch instruction from beyond the physical

memory range. This may then lead to error conditions and alarms being triggered by the bus and memory systems.

It is therefore recommended that the upper 64 bytes of any memory are not used for instruction storage.

6.6.9 Atomicity of Data Accesses

The data alignment rules along with the number of bus transactions for each access type are detailed in the following tables: -

Alignment Rules

Table 6-7 Alignment rules for non-peripheral space

Access type	Access size	Alignment of address in memory	Min/Max number of SRI bus transactions
Load, Store Data Register	Byte	Byte (1_H)	1/1
	Half-Word	2 bytes (2_H)	1/1
	Word	2 bytes (2_H)	1/2 *
	Double-Word	2 bytes (2_H)	1/2 *
Load, Store Address Register	Word	4 bytes (4_H)	1/1
	Double-Word	4 bytes (4_H)	1/2 *
SWAP.W, LDMST, CMPSWAP.W, SWAPMSK.W	Word	4 bytes (4_H)	1/1
ST.T	Byte	Byte (1_H)	1/1
Context Operations	16 x 32-bit registers	64 bytes (40_H)	2/2 *

Table 6-8 Alignment rules for peripheral space

Access type	Access size	Alignment of address in memory	Min/Max Number of SPB bus transactions	Min/Max Number of SRI bus transactions
Load, Store Data Register	Byte	Byte (1_H)	1/1	1/1
	Half-Word	2 bytes (2_H)	1/1	1/1
	Word	4 bytes (4_H)	1/1	1/1
	Double-Word	8 bytes (8_H)	1/1	1/1
Load, Store Address Register	Word	4 bytes (4_H)	1/1	1/1
	Double-Word	8 bytes (8_H)	1/1	1/1
SWAP.W, LDMST, CMPSWAP.W, SWAPMSK.W	Word	4 bytes (4_H)	1/1	1/1
ST.T	Byte	Bytes (1_H)	1/1	1/1
Context Operations	16 x 32-bit registers	Not Permitted	-	-

In the case where a single access results in a single bus transaction atomicity of the data is preserved when viewed from any bus master.

In the case where a single access leads to multiple bus transactions (marked as “*” in the above tables) then atomicity needs to be considered. In these accesses it is possible for another bus master to read or write the target memory location between the bus transactions required to complete the access.

In the case of word and double word access to the SRI bus the number of bus transactions will be 1 for naturally aligned data values and hence atomicity will be preserved.

6.6.10 A11 usage

In normal usage A11 will always contain the target of the next RET or RFE instruction. The processor uses this fact to speculatively load the return target ahead of the execution of the RET/RFE instruction. Code that modifies the A11 (e.g. test code) should be aware that any value stored in A11 may be used as the target of such speculation. If the value in A11 is not a valid address the speculation may lead to error conditions and alarms being triggered by the bus and memory systems.

It is therefore recommended that A11 should only ever contain a valid address value.

6.7 Memory Addressing

This chapter details the CPU specific addressing.

6.7.1 CSFR and SFR base Locations

Each CPU has a dedicated set of control and status registers accessed at the addresses detailed in the following table.These registers are divided into Special Function Registers (SFRs) and Core Special Function Registers (CSFRs).

A CPU must access its own CSFR registers using MTCR and MFCR instructions. CSFR registers of other CPUs may be accessed using load and store instructions via the XBAR_SRI.

SFR registers of any CPU may only be accessed using load and store instructions via XBAR_SRI. Currently the overlay control and the access protection registers of CPUx are mapped into CPUx SFR address range.

The base locations for the TC1.6P and TC1.6E SFR and CSFR registers are as follows:-

Table 6-9 CSFR and SFR Base Locations

Core-ID Value	CSFR Base Address	SFR Base Address
0	$F881_0000_H$	$F880_0000_H$
1	$F883_0000_H$	$F882_0000_H$
2	$F885_0000_H$	$F884_0000_H$

6.7.2 Local and Global Addressing

The TriCore architecture supports closely coupled program and data SRAM memories known as Program Scratch Pad RAM (PSPR) and Data Scratch Pad RAM (DSPR). The local PSPR memory is always located at $C0000000_H$. The local DSPR is always located at $D0000000_H$. In a multiprocessor system the local scratch pad memories appear in the global address map in the following locations:-

Table 6-10 Global Address Locations

Core-ID Value	PSPR_base	DSPR_base
0	70100000_H	70000000_H
1	60100000_H	60000000_H
2	50100000_H	50000000_H

The CPUs always use the global addresses for bus transactions. Thus a data load from $C0000000_H$ (a CPUs local PSPR) will result in a bus transaction with an address in the range 50100000_H - $701FFFFF_H$ dependent on the Core-ID value of the processor.

Similarly a code fetch form $D0000000_H$ (a CPUs local DSPR) will result in a bus transaction with an address in the range 50000000_H - $700FFFFF_H$ dependent on the Core-ID value of the processor.

6.7.3 Cache Memory Access

The cache and tag memories may be mapped into the CPUs address space. When mapped the cache memories are contiguous with the DSPR/PSPR memories as detailed in the following table.When mapped the cache memories behave identically to the PSPR/DSPR memories and may be used as standard memory.

The mapping of cache and tag memories to the TriCore address space is controlled by the MTU_MEMMAP register. See the MTU chapter for details.

Table 6-11 Cache Memory Locations when mapped

Memory	Local Address	Global Address
Program Cache	$C000_0000_H$+ PSPR_Memory_Size	PSPR_Base + PSPR_Memory_Size
Data Cache (TC1.6P)	$D000_0000_H$+ DSPR_Memory_Size	DSPR_Base + DSPR_Memory_Size

When mapped the tag memories are available at the locations detailed in the following table. The mapping of the Tag memories is provided test purposes only. They may not be used as standard memory.

Table 6-12 Tag Memory Locations when mapped

Memory	Local Address	Global Address
Program Tag	$C00C_0000_H$	PSPR_Base + $0x000C_0000_H$
Data Tag (TC1.6P)	$D00C_0000_H$	DSPR_Base + $0x000C_0000_H$

The CACHEI.* instructions require a way and index value to be supplied in a valid address. The location of these bits in the 32bit address is as follows.

Table 6-13 Way and Index Location

Function	Address Bits
Way	[0]
Index	[11:5]

6.8 CPU Subsystem Registers

This section describes the implementation-specific features of the CPU Subsystem registers listed in **Table 6-14**. For complete descriptions of all registers refer to the TriCore Architecture Manual.

Table 6-14 CPU Subsystem Registers

Registers	Purpose	Description
CPU Core Special Function Registers (CSFRs)	Program state information, context and stack management, interrupt and trap control, system control	see **Page 6-28**
CPU General Purpose Registers (GPRs)	General Purpose Address and Data Registers	see **Page 6-40**
CPU Memory Protection Registers (CSFRs)	Memory protection control and mode selection	see **Page 6-40**
FPU Registers (CSFRs)	Support for the standard floating point instructions.	see **Page 6-41**
Memory Integrity Registers (CSFRs)	Integrity and Protection Core Special Function Registers.	see **Page 6-42**
Core Debug Registers (CSFRs)	Debug control	see **Page 6-53**
Implementation Specific Reset Values	Reset values for CPU registers not defined in this chapter	see **Page 6-56**
Program Memory Interface Registers (PMI CSFRs)	PMI program cache control, status and trap information	see **Page 6-93**
Data Memory Interface Registers (DMI CSFRs)	DMI data cache control, status and trap information	see **Page 6-105**

Note: To increase performance all stores to CSFR or SFR register locations are posted writes and complete silently. Stores to non-existent CSFR or SFR locations are not errored.

6.8.1 CPU Core Special Function Registers (CSFR)

This section describes implementation specific features of the Core Special Function Registers.

6.8.1.1 Registers

The implementation-specific Program Status Word Register (PSW) is an extension of the PSW description in the TriCore Architecture Manual. The status flags used for FPU operations overlay the status flags used for Arithmetic Logic Unit (ALU) operations.

Program Status Word Register

Note: The non-shaded areas in the register description define the implementation-specific bits/bit fields. The shaded areas are defined in the TriCore Architecture Manual.

PSW

Program Status Word Register (CSFR_Base + FE04$_H$) Reset Value: 0000 0B80$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C or FS	**V or FI**	**SV or FV**	**AV or FZ**	**SAV or FU**	**FX**	**RM**		**0**							
rwh	rwh	rwh	rwh	rwh	rwh	rw		r							

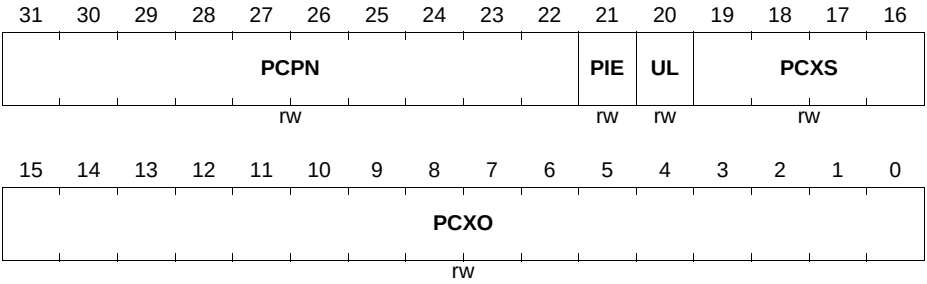
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**S**	**PRS**		**IO**		**IS**	**GW**	**CDE**	**CDC**						
r	rwh	rwh		rwh		rwh	rwh	rwh	rwh						

Field	Bits	Type	Description
RM	[25:24]	rw	**FPU Rounding Mode Selection**
FX	26	rwh	**FPU Inexact Flag**
SAV	27	rh	**Sticky Advance Overflow Flag**
FU		rwh	**FPU Underflow Flag**
AV	28	rwh	**Advance Overflow Flag**
FZ			**FPU Divide by Zero Flag**
SV	29	rwh	**Sticky Overflow Flag**
FV			**FPU Overflow Flag**

Field	Bits	Type	Description
V	30	rwh	**Overflow Flag**
FI			**FPU Invalid Operation Flag**
C	31	rwh	**Carry Flag**
FS			**FPU Some Exception Flag**

Previous Context Information Register

PCXI
Previous Context Information Register
(CSFR_Base + FE00$_H$) **Reset Value: 0000 0000$_H$**

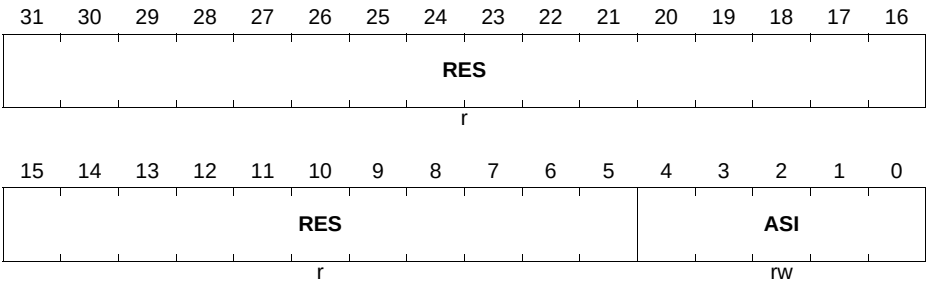


Field	Bits	Type	Description
PCXO	[15:0]	rw	**Previous Context Pointer Offset Field** The PCXO and PCXS fields form the pointer PCX, which points to the CSA of the previous context.
PCXS	[19:16]	rw	**Previous Context Pointer Segment Address** Contains the segment address portion of the PCX. This field is used in conjunction with the PCXO field.
UL	20	rw	**Upper or Lower Context Tag** Identifies the type of context saved. If the type does not match the type expected when a context restore operation is performed, a trap is generated. 0_B Lower Context 1_B Upper Context
PIE	21	rw	**Previous Interrupt Enable** Indicates the state of the interrupt enable bit (ICR.IE) for the interrupted task.
PCPN	[31:22]	rw	**Previous CPU Priority Number** Contains the priority level number of the interrupted task.

Address Space Identifier Register (TASK_ASI)

The Task Address Space Identifier (ASI) register description.

TASK_ASI
Task Address Space Identifier Register
(CSFR_Base + 8004_H) **Reset Value: $0000\ 001F_H$**

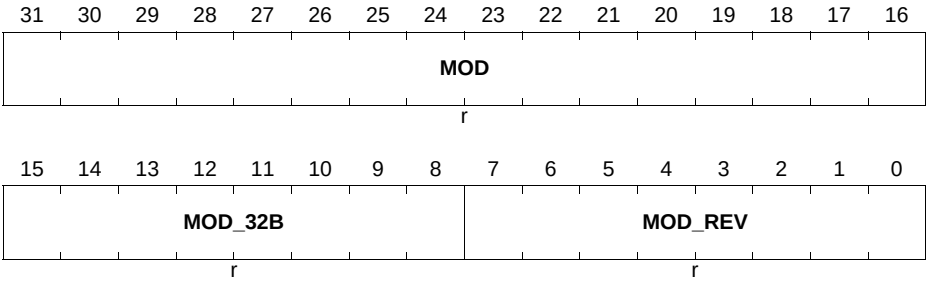


Field	Bits	Type	Description
RES	[31:5]	r	**Reserved**
ASI	[4:0]	rw	**Address Space Identifier** The ASI register contains the Address Space Identifier of the current process.

CPU Identification Register (TC1.6P)

CPU_ID
CPU Identification Register

(CSFR_Base + FE18$_H$) **Reset Value: 00C0 C012$_H$**

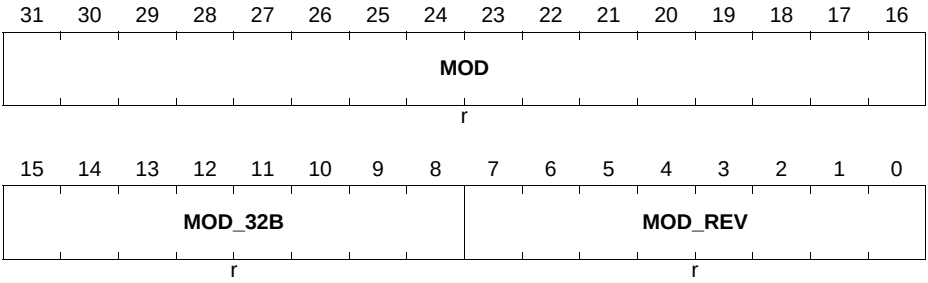


Field	Bits	Type	Description
MOD_REV	[7:0]	r	**Revision Number** 12$_H$ Reset value
MOD_32B	[15:8]	r	**32-Bit Module Enable** C0$_H$ A value of C0$_H$ in this field indicates a 32-bit module with a 32-bit module ID register.
MOD	[31:16]	r	**Module Identification Number** 00C0$_H$ For module identification

CPU Identification Register (TC1.6E)

CPU_ID
CPU Identification Register

(CSFR_Base + FE18$_H$) **Reset Value: 00B7 C002$_H$**

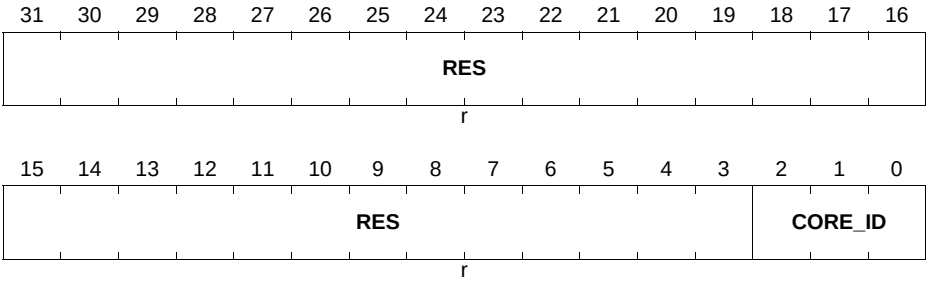


Field	Bits	Type	Description
MOD_REV	[7:0]	r	**Revision Number** 02$_H$ Reset value
MOD_32B	[15:8]	r	**32-Bit Module Enable** C0$_H$ A value of C0$_H$ in this field indicates a 32-bit module with a 32-bit module ID register.
MOD	[31:16]	r	**Module Identification Number** 00B7$_H$ For module identification

Core Identification Register

CORE_ID
Core Identification Register

(CSFR_Base + FE1C$_H$) **Reset Value: 0000 000X$_H$**

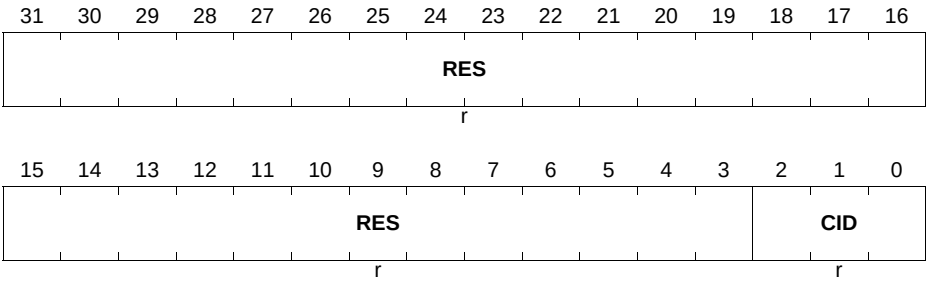


Field	Bits	Type	Description
RES	[31:3]	r	**Reserved**
CORE_ID	[2:0]	r	**Core Identification Number** The identification number of the core.

Customer ID Register

In circumstances where the Infineon defined Core identification numbering scheme is insufficient or incompatible with a customer numbering scheme a customer ID value may be supplied via the CUS_ID register.

CUS_ID
Customer ID register **(CSFR_Base + FE50$_H$)** **Reset Value: 0000 000X**



Field	Bits	Type	Description
RES	[31:3]	r	**Reserved**
CID	[2:0]	r	**Customer ID**

Physical Memory Attributes Registers

The Physical Memory Attributes registers (PMA0,PMA1,PMA2) define the physical memory attribute for each segment in the physical address space. The register is ENDINIT protected and can be read with the MFCR instruction and written by the MTCR instruction. Note that when changing the value of the registers both the instruction and data caches should be invalidated, a DSYNC instruction should be executed immediately prior to the MTCR with an ISYNC instruction executed immediately following. This is required to maintain coherency of the processors view of memory.

The register PMA0 defines the data access cacheability of the segment in the physical address space. If bit n in the register is set then segment-n will be seen as cacheable for data accesses.

The register PMA1 defines the code access cacheability of the segment in the physical address space. If bit n in the register is set then segment-n will be seen as cacheable for code accesses.

The register PMA2 defines the peripheral space identifier of the segment in the physical address space. If bit n in the register is set then segment-n will be seen as a peripheral segment.PMA2 is a read-only register.

Registers PMA0 and PMA1 are freely programmable with the following restrictions:-

- In PMA0 Segment-C and Segment[7-CoreID] must have the same value.
- In PMA1 Segment-D and Segment[7-CoreID] must have the same value.

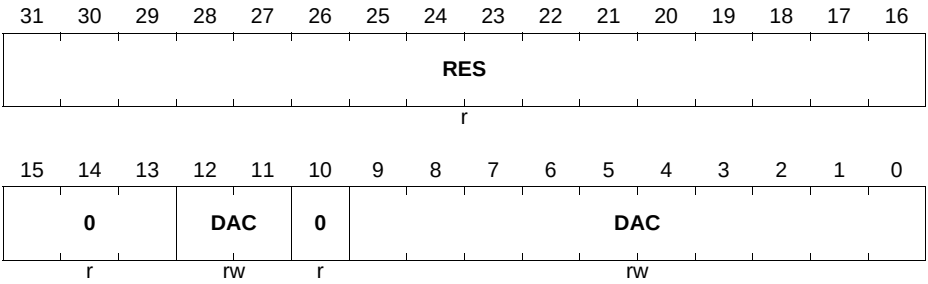
Irrespective of the setting of the PMA registers the following constraints are always enforced.

- Segments-F and segment-E are constrained to be Peripheral space and hence non-cacheable.
- Segment-A is constrained to be non-cacheable memory
- Segment-D and Segment[7-CoreID] are constrained to be non-cacheable for data accesses.
- Segment-C and Segment[7- CoreID] are constrained to be non-cacheable for code accesses.

Note: The PMA registers are ENDINIT protected.

PMA0
Data Access Cacheability Register

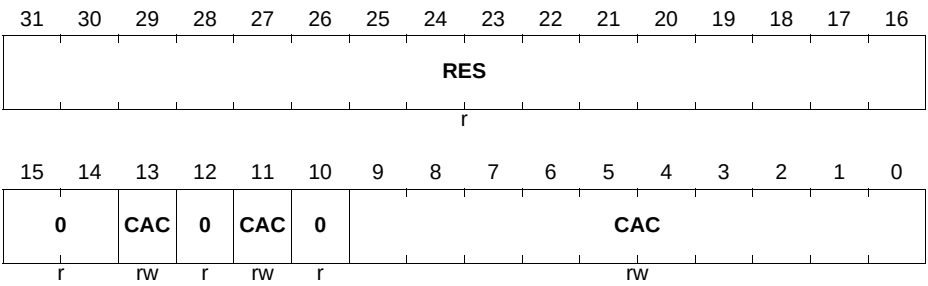
(CSFR_Base + 8100_H) **Reset Value: $0000\ 0300_H$**



Field	Bits	Type	Description
RES	[31:16]	r	**Reserved**
DAC	[15:13]	r	**Data Access Cacheability Segments F_H,E_H,D_H** (non-cacheable)
DAC	[12:11]	rw	**Data Access Cacheability Segments C_H,B_H**
DAC	10	r	**Data Access Cacheability Segments A_H** (non-cacheable)
DAC	[9:0]	rw	**Data Access Cacheability Segments 9_H-0_H**

PMA1
Code Access Cacheability Register

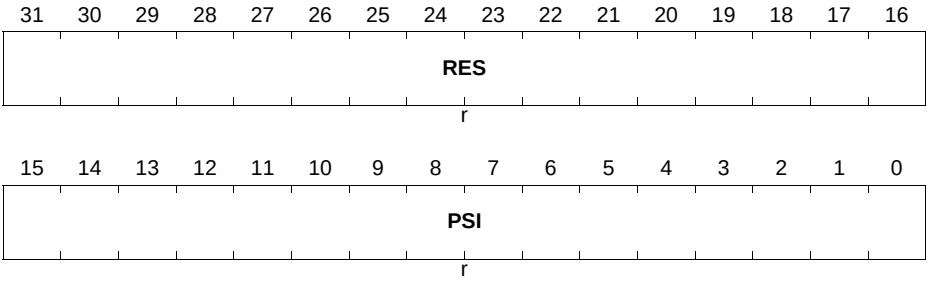
(CSFR_Base + 8104_H) **Reset Value: $0000\ 0300_H$**



Field	Bits	Type	Description
RES	[31:16]	r	**Reserved**
CAC	[15:14]	r	**Code Access Cacheability Segments F_H,E_H** (non-cacheable)
CAC	13	rw	**Code Access Cacheability Segments D_H**
CAC	12	r	**Code Access Cacheability Segments C_H** (non-cacheable)
CAC	11	rw	**Code Access Cacheability Segments B_H**
CAC	10	r	**Code Access Cacheability Segments A_H** (non-cacheable)
CAC	[9:0]	rw	**Code Access Cacheability Segments 9_H-0_H**

PMA2
Peripheral Space Identifier Register

(CSFR_Base + 8108_H) **Reset Value: 0000 $C000_H$**



Field	Bits	Type	Description
RES	[31:16]	r	**Reserved**
PSI	[15:0]	r	**Peripheral Attribute Segments F_H-0_H** = $C000_H$

Compatibility Control Register

The Compatibility Control Register (COMPAT) is an implementation-specific CSFR which allows certain elements of backwards compatibility with TriCore 1.3.x behavior to be forced. The reset value of the COMPAT register ensures that backwards compatibility with TriCore 1.3 is enabled by default. This register is safety_endinit protected.

COMPAT
Compatibility Control Register

(CSFR_Base + 9400_H) **Reset Value: FFFF FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES											SP	RM	RES		
r											rw	rw	r		

Field	Bits	Type	Description
RES	[2:0]	r	**Reserved** Read as 1; should be written with 1.
RM	3	rw	**Rounding Mode Compatibility** 0_B PSW.RM not restored by RET. 1_B PSW.RM restored by RET (TC1.3 behavior).
SP	4	rw	**SYSCON Safety Protection Mode Compatibility** 0_B SYSCON[31:1] safety endinit protected. 1_B SYSCON[31:1] not safety endinit protected (TC1.3 behavior).
RES	[31:5]	r	**Reserved** Read as 1; should be written with 1.

6.8.2 CPU General Purpose Registers

There are no implementation specific features of the General Purpose Registers. They are described in detail in the TriCore Architecture Manual.

6.8.3 CPU Memory Protection Registers

There are four Memory Protection Register Sets per CPU. The sets specify memory protection ranges and permissions for code and data. The PSW.PRS bit field determines which of these sets is currently in use by the CPU. The CPUs each implement 16 data and 8 code range comparators. These may be flexibly shared amongst the of protection sets to provide a maximum of 16 data ranges and 8 code ranges per set.

The protection registers protect the whole address space (In previous TriCore implementations the memory protection system did not cover peripheral space.) The granularity of the protection ranges is 8 bytes.

Code protection ranges define which memory areas the CPU may fetch instructions from. Instruction fetches from an area outside a valid code protection range will lead to a trap condition.

Data protection ranges define which memory areas the CPU may access for read and/or write operations. Each range may be separately enabled for read and/or write access. Data read accesses from an area outside a valid data protection range with read permissions will lead to a trap condition. Data write accesses to an area outside a valid data protection range with write permissions will lead to a trap condition

There are no implementation specific features of the Memory Protection Registers. They are described in detail in the TriCore Architecture Manual.

6.8.4 Temporal Protection Registers

To Guard against task runtime overrun the CPU implements a temporal protection system. This system consists of three independent decrementing counters (TPS_TIMERn) arranged to generate a Temporal Asynchronous Error trap (TAE - Class-4, Tin-7) on decrement to zero. A control register (TPS_CON) contains status and control bits for temporal protection system. The temporal protection system is enabled by the SYSCON.TPROTEN register bit. The Temporal Protection Registers are Core Special Function Registers. They are described in detail in the TriCore Architecture Manual.

There are no implementation specific features of the Memory Protection Registers. They are described in detail in the TriCore Architecture Manual.

6.8.5 FPU Registers

There are no implementation specific features of the FPU Registers. They are described in detail in the TriCore Architecture Manual.

6.8.6 Memory Integrity Registers

To monitor and debug the integrity of the memory subsystems the following registers are provided.

Table 6-15 Memory Integrity Registers

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
PIEAR	Program Integrity Error Address Register	9210_H	U, SV, 32	SV, 32,P	Application $0000\ 0000_H$
PIETR	Program Integrity Error Trap Register	9214_H	U, SV, 32	SV, 32,P	Application $0000\ 0000_H$
DIEAR	Data Integrity Error Address Register	9020_H	U, SV, 32	SV, 32,P	Application $0000\ 0000_H$
DIETR	Data Integrity Error Trap Register	9024_H	U, SV, 32	SV, 32,P	Application $0000\ 0000_H$
SMACON	SIST Mode Access Control Register	$900C_H$	U, SV, 32	SV, SE, 32,P	Application $0000\ 0000_H$

6.8.6.1 Register Descriptions

Program Integrity Error Information Registers

Two architecturally visible registers (PIETR, PIEAR) allow software to localise the source of the last detected program memory integrity error.

These registers are updated when a program integrity error condition is detected and the PIETR.IED bit is zero. On update the PIETR.IED bit is set to one and remains set until cleared by software. Whilst PIETR.IED is set further hardware updates of PIETR and PIEAR are inhibited. The various error scenarios are defined below.

Program Integrity errors during instruction fetch from program memory

When an error is detected during a program fetch from a program memory the IE_S, IE_C bits are updated to denote in which memory structure the error was detected. If the IE_C bit is set the E_INFO field will be updated with the cache way. The PIEAR register is updated with the address of the access. If the error detected is an uncorrectable bit error the IE_UNC bit is set. The IED bit is set and all other PIETR register bits are set to zero. Since instruction fetches are speculative, the PIETR and PIEAR registers may be updated without a corresponding PIE trap

Program Integrity errors during a memory read initiated by an external access

When an error is detected during an external bus read from program memory the IE_S, IE_C bits are updated to denote in which memory structure the error was detected. The IE_BS is set and E_INFO updated. The PIEAR register is updated with the address of the access. If the error detected is an uncorrectable error the IE_UNC bit is set. The IED bit is set and all other PIETR register bits are set to zero. Note that a memory read can be generated by a sub word write operation. The IE_C bit can only ever be set is the cache is mapped into memory in SIST mode. In this case the E_INFO field indicates the tag of the requesting master.

Program Integrity errors due to safety protection

When a safety protection violation is detected during a program memory access or during access to the CPU registers the IE_SP, IE_BS bits are set and the E_INFO field updated. The PIEAR register is updated with the address of the access. The IED bit is set and all other PIETR register bits are set to zero.

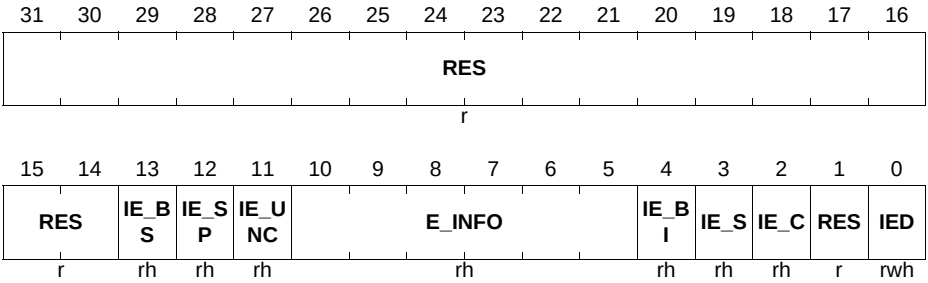
Program Integrity errors due ECC errors at the bus interface

When an ECC error is detected at the program bus interface (either master or slave) the IE_BI bit is set. If the error is during an external access the IE_BS bit is set and the E_INFO fields updated. The PIEAR register is updated with the address of the access.

If the error detected is an uncorrectable error the IE_UNC bit is set. The IED bit is set and all other PIETR register bits are set to zero.

Program Integrity Error Trap Register (PIETR)

PIETR
Program Integrity Error Trap Register
(CSFR_Base + 9214_H) **Reset Value: $0000\ 0000_H$**

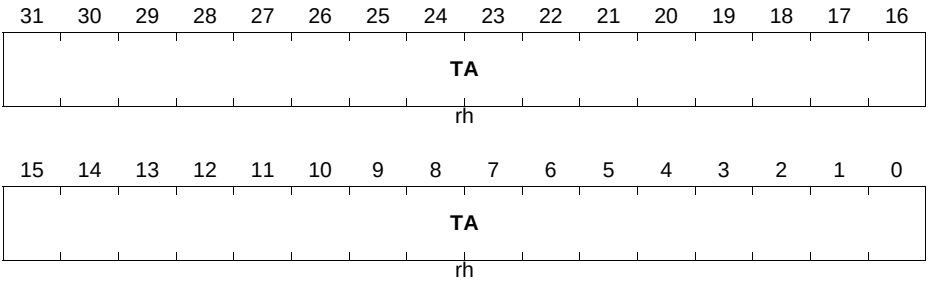


Field	Bits	Type	Description
IED	0	rwh	**Integrity Error Detected** Read Operation: 0_B No program integrity error condition occurred. 1_B Program integrity error condition detected. PIETR and PIEAR contents valid, further updates disabled. Write Operation: 0_B Clear IED bit, re-enable updates. 1_B No effect.
RES	1	r	**Reserved**
IE_C	2	rh	**Integrity Error - Cache Memory**
IE_S	3	rh	**Integrity Error - Scratchpad Memory**
IE_BI	4	rh	**Integrity Error - Bus Interface**
E_INFO	[10:5]	rh	**Error Information** If IE_BS= 1: Bus Master Tag ID of requesting master If IE_C = 1: Cache way.
IE_UNC	11	rh	**Integrity Error - Uncorrectable Error Detected**
IE_SP	12	rh	**Safety Protection Error Detected**
IE_BS	13	rh	**Bus Slave Access Indicator**
RES	[31:14]	r	**Reserved** Read as 0; should be written with 0.

Program Integrity Error Address Register

This register contains the physical address accessed by the operation that encountered a uncorrectable program memory integrity error. This register is only updated if PIETR.IED is zero.

PIEAR
Program Integrity Error Address Register
(CSFR_Base + 9210_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
TA	[31:0]	rh	**Transaction Address** Physical address being accessed by operation that encountered program integrity error.

Data Integrity Error Information Registers

Two architecturally visible registers (DIETR, DIEAR) allow software to localise the source of the last detected uncorrectable data memory integrity error.

These registers are updated when an uncorrectable data integrity error condition is detected and the DIETR.IED bit is zero. On update the DIETR.IED bit is set to one and remains set until cleared by software. Whilst DIETR.IED is set further hardware updates of DIETR and DIEAR are inhibited. The various error scenarios are defined below.

Note that the TC1.6E has no data cache but will report Data Read Buffer errors as cache errors from way0.

Data Integrity errors during load from data memory

When an error is detected during a load from a data memory the IE_S, IE_C and IE_T bits are updated to denote in which memory structure the error was detected. If the IE_C bit is set the E_INFO field will be updated with the cache way. The DIEAR register is updated with the address of the access. If the error detected is an uncorrectable error the IE_UNC bit is set. The IED bit is set and all other DIETR register bits are set to zero. Note that a memory read may also be generated by a sub word write operation or a cache line eviction.

Data Integrity errors during a memory read initiated by an external access

When an error is detected during an external bus read from data memory the IE_S, IE_C bits are updated to denote in which memory structure the error was detected. The IE_BS is set and E_INFO updated. The DIEAR register is updated with the address of the access. If the error detected is an uncorrectable error the IE_UNC bit is set. The IED bit is set and all other DIETR register bits are set to zero. Note that a memory read may also be generated by a sub word write operation or a cache line eviction. The IE_C bit can only ever be set is the cache is mapped into memory in SIST mode. In this case the E_INFO field indicates the tag of the requesting master.

Data Integrity errors due to safety protection

When a safety protection violation is detected during a data memory access the IE_SP, IE_BS bits are set and the E_INFO field updated. The DIEAR register is updated with the address of the access. The IED bit is set and all other DIETR register bits are set to zero.

Data Integrity errors due ECC errors at the bus interface

When an ECC error is detected at the data bus interface (either master or slave) the IE_BI bit is set. If the error is during an external access the IE_BS bit is set and the E_INFO fields updated. The DIEAR register is updated with the address of the access.

If the error detected is an uncorrectable error the IE_UNC bit is set. The IED bit is set and all other DIETR register bits are set to zero.

Data Integrity Error Trap Register (DIETR)

DIETR
Data Integrity Error Trap Register

(CSFR_Base + 9024_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES		IE_BS	IE_SP	IE_UNC	E_INFO						IE_BI	IE_S	IE_C	IE_T	IED
r		rh	rh	rh	rh						rh	rh	rh	rh	rwh

Field	Bits	Type	Description
IED	0	rwh	**Integrity Error Detected** Read Operation: 0_B No data integrity error condition occurred. 1_B Data integrity error condition detected. DIETR and DIEAR contents valid, further DIETR and DIEAR updates disabled. Write Operation: 0_B Clear IED bit, re-enable DIETR and DIEAR update. 1_B No effect.
IE_T	1	rh	**Integrity Error - Tag Memory**
IE_C	2	rh	**Integrity Error - Cache Memory**
IE_S	3	rh	**Integrity Error - Scratchpad Memory**
IE_BI	4	rh	**Integrity Error - Bus Interface**
E_INFO	[10:5]	rh	**Error Information** If IE_BS = 1: Bus Master Tag ID of requesting master If IE_C = 1: Cache way.
IE_UNC	11	rh	**Dual Bit Error Detected**
IE_SP	12	rh	**Safety Protection Error Detected**

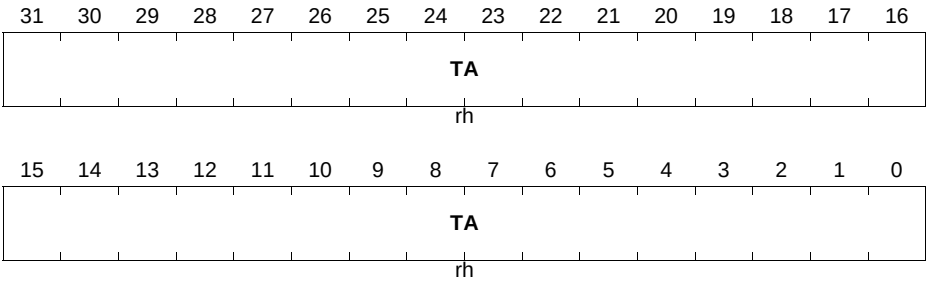
Field	Bits	Type	Description
IE_BS	13	rh	**Bus Slave Access Indicator**
RES	[31:14]	r	**Reserved** Read as 0; should be written with 0.

Data Integrity Error Address Register

This register contains the physical address accessed by the operation that encountered a uncorrectable data memory integrity error. This register is only updated if DIETR.IED is zero.

DIEAR
Data Integrity Error Address Register

(CSFR_Base + 9020_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
TA	[31:0]	rh	**Transaction Address** Physical address being accessed by operation that encountered data integrity error.

SIST (Software In-System) Test Support

The CPU protects against memory integrity errors by ECC protection of the local CPU memories. This has the side-effect of requiring memory blocks wider than the normal data access path to the memory. The additional ECC storage bits are not easily accessible via the existing data paths, causing problems where software based testing of the memories is required. The CPU memories also include embedded memory arrays, such as the tag memories, which are not ordinarily accessible by the usual CPU datapaths. In order to address this problem, the CPUs include a modes allowing all local memory arrays to be accessed, both as a backup for MBIST based memory test and to allow the test and debug of the fault tolerant memory systems.

Each memory may be access either in normal operation, data array only or ECC check array only. This is controlled by the ECCS registers of the associated MBIST controller.

The IODT bit controls read/write operation ordering. In normal operation (IODT=0) non-dependent read operations may overtake write operations. When SMA.IODT is set all memory operations are performed in program order.

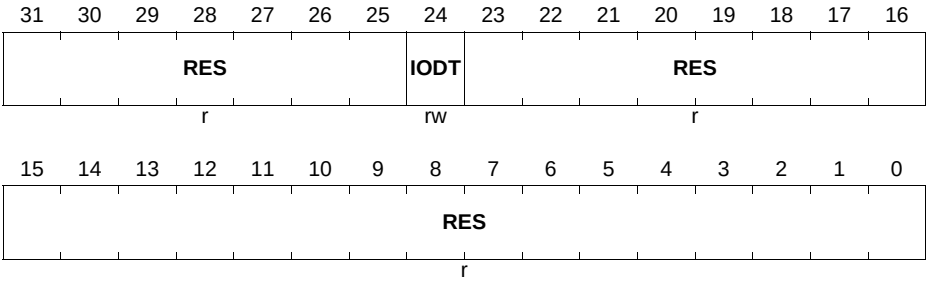
The SMACON register is protected by the safety_endinit signal.

The mapping of cache and tag memories to the TriCore address space is controlled by the MTU_MEMMAP register. See the MTU chapter for details.

SIST Mode Access Control Register

SMACON
SIST Mode Access Control Register

(CSFR_Base + 900C$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RES	[31:25]	r	**Reserved** Read as 0; should be written with 0.
IODT	24	rw	**In-Order Data Transactions** 0_B Normal operation, Non-dependent loads bypass stores. 1_B In-order operation, Loads always flush preceding stores, processor store buffer disabled.
RES	[23:0]	r	**Reserved** Read as 0; should be written with 0.

6.8.7 CPU Core Debug and Performance Counter Registers

The CPU Core Debug and performance counter registers are available for debug purposes. For a complete description of all registers, refer to the TriCore Architecture Manual.

6.8.7.1 Counter Source Details

Details of the Performance counter sources is given below.

IP_DISPATCH_STALL

The counter is incremented on every cycle in which the Integer dispatch unit is stalled for whatever reason.

LS_DISPATCH_STALL

The counter is incremented on every cycle in which the Load-Store dispatch unit is stalled for whatever reason.

LP_DISPATCH_STALL

The counter is incremented on every cycle in which the Loop dispatch unit is stalled for whatever reason.

PCACHE_HIT

The counter is incremented whenever the target of a cached fetch request from the fetch unit is found in the program cache.

PCACHE_MISS

The counter is incremented whenever the target of a cached fetch request from the fetch unit is not found in the program cache and hence a bus fetch is initiated.

MULTI_ISSUE

The counter is incremented in any cycle where more than one instruction is issued.

DCACHE_HIT

The counter is incremented whenever the target of a cached request from the Load-Store unit is found in the data cache.

DRB_HIT

The counter is incremented whenever the target of a cached request from the Load-Store unit is found in the data read buffer.

DCACHE_MISS_CLEAN

The counter is incremented whenever the target of a cached request from the Load-Store unit is not found in the data cache and hence a bus fetch is initiated with no dirty cache line eviction.

DRB_MISS

The counter is incremented whenever the target of a cached request from the Load-Store unit is not found in the data read buffer and hence a bus fetch is initiated.

DCACHE_MISS_DIRTY

The counter is incremented whenever the target of a cached request from the Load-Store unit is not found in the data cache and hence a bus fetch is initiated with the writeback of a dirty cache line.

TOTAL_BRANCH

The counter is incremented in any cycle in which a branch instruction is in a branch resolution stage of the pipeline (IP_EX1, LS_DEC, LP_DEC).

PMEM_STALL

The counter is incremented whenever the fetch unit is requesting an instruction and the Instruction memory is stalled for whatever reason.

DMEM_STALL

The counter is incremented whenever the Load-Store unit is requesting a data operation and the data memory is stalled for whatever reason.

Performance Counter Registers

Table 6-16 OCDS Control Registers

Register	Description	Offset Address
CCTRL	Counter Control Register.	$FC00_H$
CCNT	CPU Clock Count Register.	$FC04_H$
ICNT	Instruction Count Register.	$FC08_H$
M1CNT	Multi Count Register 1.	$FC0C_H$
M2CNT	Multi Count Register 2.	$FC10_H$

Table 6-16 OCDS Control Registers (cont'd)

Register	Description	Offset Address
M3CNT	Multi Count Register 3.	$FC14_H$

Table 6-17 MultiCount Configuration (TC1.6P)

CFG	M1CNT	M2CNT	M3CNT
000	IP_DISPATCH_STALL	LS_DISPATCH_STALL	LP_DISPATCH_STALL
001	PCACHE_HIT	PCACHE_MISS	MULTI_ISSUE
010	DCACHE_HIT	DCACHE_MISS_CLEAN	DCACHE_MISS_DIRTY
011	TOTAL_BRANCH	PMEM_STALL	DMEM_STALL

Table 6-18 MultiCount Configuration (TC1.6E)

CFG	M1CNT	M2CNT	M3CNT
000	IP_DISPATCH_STALL	LS_DISPATCH_STALL	-
001	PCACHE_HIT	PCACHE_MISS	-
010	DRB_HIT	DRB_MISS	-
011	TOTAL_BRANCH	PMEM_STALL	DMEM_STALL

6.8.8 Summary of CSFR Reset Values and Access Modes

This section summarizes the reset values and access modes of the CPU CSFR registers.

Table 6-19 Core Special l Function Registers

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
TASK_ASI	TASK Address Space Identifier	8004_H	U, SV, 32	SV,32, P	Application $0000\ 0000_H$
PCXI	Previous Context Information Register	$FE00_H$	U, SV, 32	SV,32, P	Application $0000\ 0000_H$
PSW	Program Status Word Register	$FE04_H$	U, SV, 32	SV,32, P	Application $0000\ 0B80_H$
PC	Program Counter Register	$FE08_H$	U, SV, 32	SV,32, P	Application $XXXX\ XXXX_H$
SYSCON	System Configuration Register	$FE14_H$	U, SV, 32	SV,32, P,SE	Application $0000\ 0000_H$
CPU_ID (TC1.6P)	CPU Identification Register	$FE18_H$	U, SV, 32	SV, 32 P, NC	Application $00C0\ C012_H$
CPU_ID (TC1.6E)	CPU Identification Register	$FE18_H$	U, SV, 32	SV,32, P,NC	Application $00B7\ C002_H$
CORE_ID	Core Identification Register	$FE1C_H$	U, SV, 32	SV 32, P, NC	Application $00000\ 000X_H$
BIV	Interrupt Vector Table Pointer Register	$FE20_H$	U, SV, 32	SV,32 P,CEx	Application $0000\ 0000_H$
BTV	Trap Vector Table Pointer Register	$FE24_H$	U, SV, 32	SV,32, P,CEx	Application $A000\ 0100_H$
ISP	Interrupt Stack Pointer Register	$FE28_H$	U, SV, 32	SV,32 P,CEx	Application $0000\ 0100_H$
ICR	ICU Interrupt Control Register	$FE2C_H$	U, SV, 32	SV,32, P	Application $0000\ 0000_H$
FCX	Free Context List Head Pointer Register	$FE38_H$	U, SV, 32	SV,32, P	Application $0000\ 0000_H$
LCX	Free Context List Limit Pointer Register	$FE3C_H$	U, SV, 32	SV,32, P	Application $0000\ 0000_H$
PMA0	Physical Memory Attributes Register 0	8100_H	U, SV, 32	SV,32 P, CEx	Application $0000\ 0300_H$

Table 6-19 Core Special (cont'd)I Function Registers

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
PMA1	Physical Memory Attributes Register 1	8104_H	U, SV, 32	SV,32, P, CEx	Application $0000\ 0300_H$
PMA2	Physical Memory Attributes Register 2	8108_H	U, SV, 32	SV,32, P, NC	Application $0000\ C000_H$
SMACON	SIST Mode Access Control register	$900C_H$	U, SV, 32	SV,32, P, SE	Application $0000\ 0000_H$
COMPAT	Compatibility Control Register	9400_H	U, SV, 32	SV,32 P, SE	Application $FFFF\ FFFF_H$

Table 6-20 GPR Registers

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
D0	Data Register 0	$FF00_H$	U, SV, 32	SV,32 P	Application $XXXX\ XXXX_H$
D1	Data Register 1	$FF04_H$	U, SV, 32	SV,32 P	Application $XXXX\ XXXX_H$
D2	Data Register 2	$FF08_H$	U, SV, 32	SV,32 P	Application $XXXX\ XXXX_H$
D3	Data Register 3	$FF0C_H$	U, SV, 32	SV,32 P	Application $XXXX\ XXXX_H$
D4	Data Register 4	$FF10_H$	U, SV, 32	SV,32 P	Application $XXXX\ XXXX_H$
D5	Data Register 5	$FF14_H$	U, SV, 32	SV,32 P	Application $XXXX\ XXXX_H$
D6	Data Register 6	$FF18_H$	U, SV, 32	SV,32 P	Application $XXXX\ XXXX_H$
D7	Data Register 7	$FF1C_H$	U, SV, 32	SV,32 P	Application $XXXX\ XXXX_H$
D8	Data Register 8	$FF20_H$	U, SV, 32	SV,32 P	Application $XXXX\ XXXX_H$
D9	Data Register 9	$FF24_H$	U, SV, 32	SV,32 P	Application $XXXX\ XXXX_H$

Table 6-20 GPR Registers (cont'd)

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
D10	Data Register 10	$FF28_H$	U, SV, 32	SV,32 P	Application XXXX XXXX$_H$
D11	Data Register 11	$FF2C_H$	U, SV, 32	SV,32 P	Application XXXX XXXX$_H$
D12	Data Register 12	$FF30_H$	U, SV, 32	SV,32 P	Application XXXX XXXX$_H$
D13	Data Register 13	$FF34_H$	U, SV, 32	SV,32 P	Application XXXX XXXX$_H$
D14	Data Register 14	$FF38_H$	U, SV, 32	SV,32 P	Application XXXX XXXX$_H$
D15	Data Register 15	$FF3C_H$	U, SV, 32	SV,32 P	Application XXXX XXXX$_H$
A0	Address Register 0 (Global Address Register)	$FF80_H$	U, SV, 32	SV,32 P	Application XXXX XXXX$_H$
A1	Address Register 1 (Global Address Register)	$FF84_H$	U, SV, 32	SV,32 P	Application XXXX XXXX$_H$
A2	Address Register 2	$FF88_H$	U, SV, 32	SV,32 P	Application XXXX XXXX$_H$
A3	Address Register 3	$FF8C_H$	U, SV, 32	SV,32 P	Application XXXX XXXX$_H$
A4	Address Register 4	$FF90_H$	U, SV, 32	SV,32 P	Application XXXX XXXX$_H$
A5	Address Register 5	$FF94_H$	U, SV, 32	SV,32 P	Application XXXX XXXX$_H$
A6	Address Register 6	$FF98_H$	U, SV, 32	SV,32 P	Application XXXX XXXX$_H$
A7	Address Register 7	$FF9C_H$	U, SV, 32	SV,32 P	Application XXXX XXXX$_H$
A8	Address Register 8 (Global Address Register)	$FFA0_H$	U, SV, 32	SV,32 P	Application XXXX XXXX$_H$
A9	Address Register 9 (Global Address Register)	$FFA4_H$	U, SV, 32	SV,32 P	Application XXXX XXXX$_H$

Table 6-20 GPR Registers (cont'd)

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
A10	Address Register 10 (Stack Pointer)	$FFA8_H$	U, SV, 32	SV,32 P	Application XXXX XXXX$_H$
A11	Address Register 11 (Return Address)	$FFAC_H$	U, SV, 32	SV,32 P	Application XXXX XXXX$_H$
A12	Address Register 12	$FFB0_H$	U, SV, 32	SV,32 P	Application XXXX XXXX$_H$
A13	Address Register 13	$FFB4_H$	U, SV, 32	SV,32 P	Application XXXX XXXX$_H$
A14	Address Register 14	$FFB8_H$	U, SV, 32	SV,32 P	Application XXXX XXXX$_H$
A15	Address Register 15	$FFBC_H$	U, SV, 32	SV,32 P	Application XXXX XXXX$_H$

Table 6-21 Memory Protection Registers

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
DPR0_L	Data Protection Range 0, Lower Bound Register	$C000_H$	U, SV, 32	SV,32 P	Application 0000 0000$_H$
DPR0_U	Data Protection Range 0, Upper Bound Register	$C004_H$	U, SV, 32	SV,32 P	Application 0000 0000$_H$
DPR1_L	Data Protection Range 1, Lower Bound Register	$C008_H$	U, SV, 32	SV,32 P	Application 0000 0000$_H$
DPR1_U	Data Protection Range 1, Upper Bound Register	$C00C_H$	U, SV, 32	SV,32 P	Application 0000 0000$_H$
DPR2_L	Data Protection Range 2, Lower Bound Register	$C010_H$	U, SV, 32	SV,32 P	Application 0000 0000$_H$
DPR2_U	Data Protection Range 2, Upper Bound Register	$C014_H$	U, SV, 32	SV,32 P	Application 0000 0000$_H$
DPR3_L	Data Protection Range 3, Lower Bound Register	$C018_H$	U, SV, 32	SV,32 P	Application 0000 0000$_H$
DPR3_U	Data Protection Range 3, Upper Bound Register	$C01C_H$	U, SV, 32	SV,32 P	Application 0000 0000$_H$

Table 6-21 Memory Protection Registers (cont'd)

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
DPR4_L	Data Protection Range 4, Lower Bound Register	$C020_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPR4_U	Data Protection Range 4, Upper Bound Register	$C024_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPR5_L	Data Protection Range 5, Lower Bound Register	$C028_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPR5_U	Data Protection Range 5, Upper Bound Register	$C02C_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPR6_L	Data Protection Range 6, Lower Bound Register	$C030_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPR6_U	Data Protection Range 6, Upper Bound Register	$C034_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPR7_L	Data Protection Range 7, Lower Bound Register	$C038_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPR7_U	Data Protection Range 7, Upper Bound Register	$C03C_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPR8_L	Data Protection Range 8, Lower Bound Register	$C040_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPR8_U	Data Protection Range 8, Upper Bound Register	$C044_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPR9_L	Data Protection Range 9, Lower Bound Register	$C048_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPR9_U	Data Protection Range 9, Upper Bound Register	$C04C_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPR10_L	Data Protection Range 10, Lower Bound Register	$C050_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPR10_U	Data Protection Range 10, Upper Bound Register	$C054_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPR11_L	Data Protection Range 11, Lower Bound Register	$C058_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPR11_U	Data Protection Range 11, Upper Bound Register	$C05C_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$

Table 6-21 Memory Protection Registers (cont'd)

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
DPR12_L	Data Protection Range 12, Lower Bound Register	$C060_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPR12_U	Data Protection Range 12, Upper Bound Register	$C064_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPR13_L	Data Protection Range 13, Lower Bound Register	$C068_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPR13_U	Data Protection Range 13, Upper Bound Register	$C06C_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPR14_L	Data Protection Range 14, Lower Bound Register	$C070_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPR14_U	Data Protection Range 14, Upper Bound Register	$C074_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPR15_L	Data Protection Range 15, Lower Bound Register	$C078_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPR15_U	Data Protection Range 15, Upper Bound Register	$C07C_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
CPR0_L	Code Protection Range 0, Lower Bound Register	$D000_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
CPR0_U	Code Protection Range 0, Upper Bound Register	$D004_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
CPR1_L	Code Protection Range 1, Lower Bound Register	$D008_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
CPR1_U	Code Protection Range 1, Upper Bound Register	$D00C_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
CPR2_L	Code Protection Range 2, Lower Bound Register	$D010_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
CPR2_U	Code Protection Range 2, Upper Bound Register	$D014_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
CPR3_L	Code Protection Range 3, Lower Bound Register	$D018_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
CPR3_U	Code Protection Range 3, Upper Bound Register	$D01C_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$

Table 6-21 Memory Protection Registers (cont'd)

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
CPR4_L	Code Protection Range 4, Lower Bound Register	$D020_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
CPR4_U	Code Protection Range 4, Upper Bound Register	$D024_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
CPR5_L	Code Protection Range 5, Lower Bound Register	$D028_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
CPR5_U	Code Protection Range 5, Upper Bound Register	$D02C_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
CPR6_L	Code Protection Range 6, Lower Bound Register	$D030_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
CPR6_U	Code Protection Range 6, Upper Bound Register	$D034_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
CPR7_L	Code Protection Range 7, Lower Bound Register	$D038_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
CPR7_U	Code Protection Range 7, Upper Bound Register	$D03C_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
CPXE_0	Code Protection Set-0 Execute Enable	$E000_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
CPXE_1	Code Protection Set-1 Execute Enable	$E004_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
CPXE_2	Code Protection Set-2 Execute Enable	$E008_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
CPXE_3	Code Protection Set-3 Execute Enable	$E00C_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPRE_0	Data Protection Set-0 Read Enable	$E010_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPRE_1	Data Protection Set-1 Read Enable	$E014_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPRE_2	Data Protection Set-2 Read Enable	$E018_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPRE_3	Data Protection Set-3 Read Enable	$E01C_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$

Table 6-21 Memory Protection Registers (cont'd)

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
DPWE_0	Data Protection Set-0 Write Enable	$E020_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPWE_1	Data Protection Set-1 Write Enable	$E024_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPWE_2	Data Protection Set-2 Write Enable	$E028_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DPWE_3	Data Protection Set-3 Write Enable	$E02C_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$

Table 6-22 Temporal Protection System Registers

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
TPS_CON	Control Register	$E400_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
TPS_TIMER0	Timer 0 Register	$E404_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
TPS_TIMER1	Timer 1 Register	$E408_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
TPS_TIMER2	Timer 2 Register	$E40C_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$

Table 6-23 Floating Point Special Function Registers

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
FPU_TRAP_CON	Trap Control Register	$A000_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
FPU_TRAP_PC	Trapping Instruction Program Counter Register	$A004_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
FPU_TRAP_OPC	Trapping Instruction Opcode Register	$A008_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
FPU_TRAP_SRC1	Trapping Instruction Operand Register	$A010_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$

Table 6-23 Floating Point Special Function Registers (cont'd)

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
FPU_TRAP_SRC2	Trapping Instruction Operand Register	$A014_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
FPU_TRAP_SRC3	Trapping Instruction Operand Register	$A018_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$

Table 6-24 Core Debug and Performance Counter Registers

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
CCTRL	Counter Control Register	$FC00_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
CCNT	CPU Clock Count Register	$FC04_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
ICNT	Instruction Count Register	$FC08_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
M1CNT	Multi-Count Register 1	$FC0C_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
M2CNT	Multi-Count Register 2	$FC10_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
M3CNT	Multi-Count Register 3	$FC14_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
DBGSR	Debug Status Register	$FD00_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
EXEVT	External Event Register	$FD08_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
CREVT	Core Register Access Event Register	$FD0C_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
SWEVT	Software Debug Event Register	$FD10_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
TRIG_ACC	Debug Trigger Accumulation Register	$FD30_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
DMS	Debug Monitor Start Address Register	$FD40_H$	U, SV, 32	SV,32 P	Application CPU Dependent

Table 6-24 Core Debug and Performance Counter Registers (cont'd)

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
DCX	Debug Context Save Area Pointer Register	$FD44_H$	U, SV, 32	SV,32 P	Application CPU Dependent
DBGTCR	Debug Trap Control Register	$FD48_H$	U, SV, 32	SV,32 P	Application, Debug $0000\ 0001_H$
TR0EVT	Trigger Event 0 Configuration Register	$F000_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
TR0ADR	Trigger Event 0 Address Register	$F004_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
TR1EVT	Trigger Event 1 Configuration Register	$F008_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
TR1ADR	Trigger Event 1 Address Register	$F00C_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
TR2EVT	Trigger Event 2 Configuration Register	$F010_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
TR2ADR	Trigger Event 2 Address Register	$F014_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
TR3EVT	Trigger Event 3 Configuration Register	$F018_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
TR3ADR	Trigger Event 3 Address Register	$F01C_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
TR4EVT	Trigger Event 4 Configuration Register	$F020_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
TR4ADR	Trigger Event 4 Address Register	$F024_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
TR5EVT	Trigger Event 5 Configuration Register	$F028_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
TR5ADR	Trigger Event 5 Address Register	$F02C_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
TR6EVT	Trigger Event 6 Configuration Register	$F030_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
TR6ADR	Trigger Event 6 Address Register	$F034_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$

Table 6-24 Core Debug and Performance Counter Registers (cont'd)

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
TR7EVT	Trigger Event 7 Configuration Register	$F038_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$
TR7ADR	Trigger Event 7 Address Register	$F03C_H$	U, SV, 32	SV,32 P	Debug $0000\ 0000_H$

Table 6-25 DMI Registers

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
DCON0	DMI Control Register 0	9040_H	U, SV, 32	SV,32 P,CEx	Application $0000\ 0002_H$
DCON2	DMI Control Register 2	9000_H	U, SV, 32	SV,32 P,CEx	Application CPU Dependent
DSTR	DMI Synchronous Trap Flag Register	9010_H	U, SV, 32[1)]	SV,32 P	Application $0000\ 0000_H$
DATR	DMI Asynchronous Trap Flag Register	9018_H	U, SV, 32[1)]	SV,32 P	Application $0000\ 0000_H$
DEADD	DMI Data Error Address Register	$901C_H$	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DIEAR	Data Integrity Error Address Register	9020_H	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
DIETR	Data Integrity Error Trap Register	9024_H	U, SV, 32	SV,32 P	Application $0000\ 0000_H$
SEGEN	SRI Error Generation	1030_H	U, SV, 32	SV,32 P,CEx	Application $0000\ 0000_H$

1) Writing these registers clears the contents independent of the data value.

Table 6-26 PMI Registers

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
PCON0	PMI Control Register 0	$920C_H$	U, SV, 32	SV,32 P,CEx	Application 0000 0002_H
PCON1	PMI Control Register 1	9204_H	U, SV, 32	SV,32 P	Application 0000 0000_H
PCON2	PMI Control Register 2	9208_H	U, SV, 32	SV,32 P,NC	Application CPU Dependent
PSTR	PMI Synchronous Trap Register	9200_H	U, SV, 32	SV,32 P	Application 0000 0000_H
PIEAR	Program Integrity Error Address Register	9210_H	U, SV, 32	SV,32 P	Application 0000 0000_H
PIETR	Program Integrity Error Trap Register	9214_H	U, SV, 32	SV,32 P	Application 0000 0000_H

6.8.9 Summary of SFR Reset Values and Access modes

This section summarizes the reset values and access modes of the CPU SFR registers.

Table 6-27 Safety Protection Registers

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
RGNLA0	Safety Protection Lower Range Register 0	$E000_H$	U, SV, 32	U,SV, 32,P, SE	Application 0000 0000_H
RGNUA0	Safety Protection Upper Range Register 0	$E004_H$	U, SV, 32	U,SV, 32,P, SE	Application 0000 0000_H
RGNACCENA0	Safety Protection Access Enable Register A Range 0	$E008_H$	U, SV, 32	U,SV, 32, P SE	Application 0000 0000_H
RGNACCENB0	Safety Protection Access Enable Register B Range 0	$E00C_H$	U, SV, 32	U,SV, 32, P NC	Application 0000 0000_H

Table 6-27 Safety Protection Registers (cont'd)

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
RGNLA1	Safety Protection Lower Range Register 1	$E010_H$	U, SV, 32	U,SV, 32,P, SE	Application 0000 0000_H
RGNUA1	Safety Protection Upper Range Register 1	$E014_H$	U, SV, 32	U,SV, 32,P, SE	Application 0000 0000_H
RGNACCENA1	Safety Protection Access Enable Register A Range 1	$E018_H$	U, SV, 32	U,SV, 32,P, SE	Application 0000 0000_H
RGNACCENB1	Safety Protection Access Enable Register B Range 1	$E01C_H$	U, SV, 32	U,SV, 32,P, NC	Application 0000 0000_H
RGNLA02	Safety Protection Lower Range Register 2	$E020_H$	U, SV, 32	U,SV, 32,P, SE	Application 0000 0000_H
RGNUA2	Safety Protection Upper Range Register 2	$E024_H$	U, SV, 32	U,SV, 32,P, SE	Application 0000 0000_H
RGNACCENA2	Safety Protection Access Enable Register A Range 2	$E028_H$	U, SV, 32	U,SV, 32,P, SE	Application 0000 0000_H
RGNACCENB2	Safety Protection Access Enable Register B Range 2	$E02C_H$	U, SV, 32	U,SV, 32,P, NC	Application 0000 0000_H
RGNLA03	Safety Protection Lower Range Register 3	$E030_H$	U, SV, 32	U,SV, 32,P, SE	Application 0000 0000_H
RGNUA3	Safety Protection Upper Range Register 3	$E034_H$	U, SV, 32	U,SV, 32,P, SE	Application 0000 0000_H
RGNACCENA3	Safety Protection Access Enable Register A Range 3	$E038_H$	U, SV, 32	U,SV, 32,P, SE	Application 0000 0000_H

Table 6-27 Safety Protection Registers (cont'd)

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
RGNACCENB3	Safety Protection Access Enable Register B Range 3	$E03C_H$	U, SV, 32	U,SV, 32,P, NC	Application 0000 0000_H
RGNLA04	Safety Protection Lower Range Register 4	$E040_H$	U, SV, 32	U,SV, 32,SE	Application 0000 0000_H
RGNUA4	Safety Protection Upper Range Register 4	$E044_H$	U, SV, 32	U,SV, 32,SE	Application 0000 0000_H
RGNACCENA4	Safety Protection Access Enable Register A Range 4	$E048_H$	U, SV, 32	U,SV, 32,SE	Application 0000 0000_H
RGNACCENB4	Safety Protection Access Enable Register B Range 4	$E04C_H$	U, SV, 32	U,SV, 32,NC	Application 0000 0000_H
RGNLA05	Safety Protection Lower Range Register 5	$E050_H$	U, SV, 32	U,SV, 32,SE	Application 0000 0000_H
RGNUA5	Safety Protection Upper Range Register 5	$E054_H$	U, SV, 32	U,SV, 32,SE	Application 0000 0000_H
RGNACCENA5	Safety Protection Access Enable Register A Range 5	$E058_H$	U, SV, 32	U,SV, 32,SE	Application 0000 0000_H
RGNACCENB5	Safety Protection Access Enable Register B Range 5	$E05C_H$	U, SV, 32	U,SV, 32,NC	Application 0000 0000_H
RGNLA06	Safety Protection Lower Range Register 6	$E060_H$	U, SV, 32	U,SV, 32,SE	Application 0000 0000_H
RGNUA6	Safety Protection Upper Range Register 6	$E064_H$	U, SV, 32	U,SV, 32,SE	Application 0000 0000_H
RGNACCENA6	Safety Protection Access Enable Register A Range 6	$E068_H$	U, SV, 32	U,SV, 32,SE	Application 0000 0000_H
RGNACCENB6	Safety Protection Access Enable Register B Range 6	$E06C_H$	U, SV, 32	U,SV, 32,NC	Application 0000 0000_H
RGNLA07	Safety Protection Lower Range Register 7	$E070_H$	U, SV, 32	U,SV, 32,SE	Application 0000 0000_H
RGNUA7	Safety Protection Upper Range Register 7	$E074_H$	U, SV, 32	U,SV, 32,SE	Application 0000 0000_H
RGNACCENA7	Safety Protection Access Enable Register A Range 7	$E078_H$	U, SV, 32	U,SV, 32,SE	Application 0000 0000_H

Table 6-27 Safety Protection Registers (cont'd)

Short Name	Description	Offset Address	Access Mode		Reset Type, Value
			Read	Write	
RGNACCENB7	Safety Protection Access Enable Register B Range 7	$E07C_H$	U, SV, 32	U,SV, 32,NC	Application $0000\ 0000_H$
ACCENA	Access Enable Register A	$E100_H$	U, SV, 32	U,SV, 32,SE	Application $0000\ 0000_H$
ACCENB	Access Enable Register B	$E104_H$	U, SV, 32	U,SV, 32,SE	Application $0000\ 0000_H$

Note: A disallowed access to any CPU register (e.g. attempted write to read only register, attempted user mode access to SV, attempted access to E without Endinit, etc.) will NOT result in a Bus Error.

6.9 CPU Instruction Timing

This section gives information on CPU instruction timing by execution unit. The Integer Pipeline and Load/Store Pipeline are always present, and the Floating Point Unit (FPU) is optional. The Load/Store unit implements the optional TLB instructions.

Definition of Terms:

- **Repeat Rate**

Assuming the same instruction is being issued sequentially, repeat is the minimum number of clock cycles between two consecutive issues. There may be additional delays described elsewhere due to internal pipeline effects when issuing a different subsequent instruction.

- **Result Latency**

The number of clock cycles from the cycle when the instruction is issued to the cycle when the result value is available to be used as an operand to a subsequent instruction or written into a GPR. Result latency is not meaningful for instructions that do not write a value into a GPR.

- **Address Latency**

The number of clocks cycles from the cycle when the instruction is issued to the cycle when the addressing mode updated value is available as an operand to a subsequent instruction or written into an Address Register.

- **Flow Latency**

The number of clock cycles from the cycle when the instruction is issued to the cycle when the next instruction (located at the target location or the next sequential instruction if the control change is conditional) is issued.

6.9.1 Integer-Pipeline Instructions

These are the Integer-Pipeline instruction timings for each instruction.

6.9.1.1 Simple Arithmetic Instruction Timings

Each instruction is single issued.

Table 6-28 Simple Arithmetic Instruction Timing

Instruction	Result Latency TC16P-TC16E		Repeat Rate TC16P-TC16E		Instruction	Result Latency TC16P-TC16E		Repeat Rate TC16P-TC16E	
Integer Pipeline Arithmetic Instructions									
ABS	1	1	1	1	**MAX.H**	1	1	1	1
ABS.B	1	1	1	1	**MAX.HU**	1	1	1	1
ABS.H	1	1	1	1	**MAX.U**	1	1	1	1
ABSDIF	1	1	1	1	**MIN**	1	1	1	1
ABSDIF.B	1	1	1	1	**MIN.B**	1	1	1	1
ABSDIF.H	1	1	1	1	**MIN.BU**	1	1	1	1
ABSDIFS	2	1	1	1	**MIN.H**	1	1	1	1
ABSDIFS.H	2	1	1	1	**MIN.HU**	1	1	1	1
ABSS	2	1	1	1	**MIN.U**	1	1	1	1
ABSS.H	2	1	1	1	**RSUB**	1	1	1	1
ADD	1	1	1	1	**RSUBS**	2	1	1	1
ADD.B	1	1	1	1	**RSUBS.U**	2	1	1	1
ADD.H	1	1	1	1	**SAT.B**	1	1	1	1
ADDC	1	1	1	1	**SAT.BU**	1	1	1	1
ADDI	1	1	1	1	**SAT.H**	1	1	1	1
ADDIH	1	1	1	1	**SAT.HU**	1	1	1	1
ADDS	2	1	1	1	**SEL**	1	1	1	1
ADDS.H	2	1	1	1	**SELN**	1	1	1	1
ADDS.HU	2	1	1	1	**SUB**	1	1	1	1
ADDS.U	2	1	1	1	**SUB.B**	1	1	1	1
ADDX	1	1	1	1	**SUB.H**	1	1	1	1
CADD	1	1	1	1	**SUBC**	1	1	1	1

Table 6-28 Simple Arithmetic Instruction Timing (cont'd)

Instruction	Result Latency TC16P-TC16E		Repeat Rate TC16P-TC16E		Instruction	Result Latency TC16P-TC16E		Repeat Rate TC16P-TC16E	
CADDN	1	1	1	1	**SUBS**	2	1	1	1
CSUB	1	1	1	1	**SUBS.H**	2	1	1	1
CSUBN	1	1	1	1	**SUBS.HU**	2	1	1	1
MAX	1	1	1	1	**SUBS.U**	2	1	1	1
MAX.B	1	1	1	1	**SUBX**	1	1	1	1
MAX.BU	1	1	1	1					
Compare Instructions									
EQ	1	1	1	1	**LT.B**	1	1	1	1
EQ.B	1	1	1	1	**LT.BU**	1	1	1	1
EQ.H	1	1	1	1	**LT.H**	1	1	1	1
EQ.W	1	1	1	1	**LT.HU**	1	1	1	1
EQANY.B	1	1	1	1	**LT.U**	1	1	1	1
EQANY.H	1	1	1	1	**LT.W**	1	1	1	1
GE	1	1	1	1	**LT.WU**	1	1	1	1
GE.U	1	1	1	1	**NE**	1	1	1	1
LT	1	1	1	1					
Count Instructions									
CLO	1	1	1	1	**CLS.H**	1	1	1	1
CLO.H	1	1	1	1	**CLZ**	1	1	1	1
CLS	1	1	1	1	**CLZ.H**	1	1	1	1
Extract Instructions									
DEXTR	2	1	1	1	**INS.T**	1	1	1	1
EXTR	2	1	1	1	**INSN.T**	1	1	1	1
EXTR.U	2	1	1	1	**INSERT**	2	1	1	1
IMASK	2	1	1	1					
Logical Instructions									
AND	1	1	1	1	**OR.EQ**	1	1	1	1
AND.AND.T	1	1	1	1	**OR.GE**	1	1	1	1
AND.ANDN.T	1	1	1	1	**OR.GE.U**	1	1	1	1

Table 6-28 Simple Arithmetic Instruction Timing (cont'd)

Instruction	Result Latency TC16P-TC16E		Repeat Rate TC16P-TC16E		Instruction	Result Latency TC16P-TC16E		Repeat Rate TC16P-TC16E	
AND.EQ	1	1	1	1	**OR.LT**	1	1	1	1
AND.GE	1	1	1	1	**OR.LT.U**	1	1	1	1
AND.GE.U	1	1	1	1	**OR.NE**	1	1	1	1
AND.LT	1	1	1	1	**OR.NOR.T**	1	1	1	1
AND.LT.U	1	1	1	1	**OR.OR.T**	1	1	1	1
AND.NE	1	1	1	1	**OR.T**	1	1	1	1
AND.NOR.T	1	1	1	1	**ORN**	1	1	1	1
AND.OR.T	1	1	1	1	**ORN.T**	1	1	1	1
AND.T	1	1	1	1	**XNOR**	1	1	1	1
ANDN	1	1	1	1	**XNOR.T**	1	1	1	1
ANDN.T	1	1	1	1	**XOR**	1	1	1	1
NAND	1	1	1	1	**XOR.EQ**	1	1	1	1
NAND.T	1	1	1	1	**XOR.GE**	1	1	1	1
NOR	1	1	1	1	**XOR.GE.U**	1	1	1	1
NOR.T	1	1	1	1	**XOR.LT**	1	1	1	1
OR	1	1	1	1	**XOR.LT.U**	1	1	1	1
OR.AND.T	1	1	1	1	**XOR.NE**	1	1	1	1
OR.ANDN.T	1	1	1	1	**XOR.T**	1	1	1	1
Move Instructions									
CMOV	1	1	1	1	**MOV.U**	1	1	1	1
CMOVN	1	1	1	1	**MOVH**	1	1	1	1
MOV (32 Bit)	1	1	1	1	**MOV (64bit)**	2	1	1	1
Shift Instructions									
SH	1	1	1	1	**SH.NE**	1	1	1	1
SH.AND.T	1	1	1	1	**SH.NOR.T**	1	1	1	1
SH.ANDN.T	1	1	1	1	**SH.OR.T**	1	1	1	1
SH.EQ	1	1	1	1	**SH.ORN.T**	1	1	1	1
SH.GE	1	1	1	1	**SH.XNOR.T**	1	1	1	1
SH.GE.U	1	1	1	1	**SH.XOR.T**	1	1	1	1

Table 6-28 Simple Arithmetic Instruction Timing (cont'd)

Instruction	Result Latency TC16P-TC16E		Repeat Rate TC16P-TC16E		Instruction	Result Latency TC16P-TC16E		Repeat Rate TC16P-TC16E	
SH.H	1	1	1	1	**SHA**	1	1	1	1
SH.LT	1	1	1	1	**SHA.H**	1	1	1	1
SH.LT.U	1	1	1	1	**SHAS**	2	1	1	1
SH.NAND.T	1	1	1	1					
Coprocessor 0 Instructions									
BMERGE	2	2	1	1	**IXMIN**	2	2	1	1
BSPLIT	2	2	1	1	**UNPACK**	2	2	1	1
PARITY	2	2	1	1	**IXMAX**	2	2	1	1
PACK	2	2	1	1	**IXMAX.U**	2	2	1	1
IXMIN.U	2	2	1	1	**CRC32**	2	2	1	1
Integer Divide Instructions									
DVADJ	2	2	1	1	**DVSTEP**	6	4	4	3
DVINIT	2	2	1	1	**DVSTEP.U**	6	4	4	3
DVINIT.U	2	2	1	1	**DIV**	4-11	3-10	3-9	3-9
DVINIT.B	2	2	1	1	**DIV.U**	4-11	3-10	3-9	3-9
DVINIT.H	2	2	1	1					
DVINIT.BU	2	2	1	1					
DVINIT.HU	2	2	1	1					

The latency and repeat rate values listed for the DIV and DIV.U instructions are the minimum and maximum values.The algorithm used allows for early termination of the instruction once the full result is available.

6.9.1.2 Multiply Instruction Timings

Each instruction is single issued.

Table 6-29 Multiply Instruction Timing

Instruction	Result Latency TC16P-TC16E		Repeat Rate TC16P-TC16E		Instruction	Result Latency TC16P -TC16E		Repeat Rate TC16P-TC16E	
MUL	2	2	1	1	**MUL.Q**	3	2	1	1
MUL.U	2	2	1	1	**MULM.H**	3	2	1	1
MULS	3	2	1	1	**MULR.H**	3	2	1	1
MULS.U	3	2	1	1	**MULR.Q**	3	2	1	1
MUL.H	3	2	1	1					

6.9.1.3 Multiply Accumulate (MAC) Instruction Timing

Each instruction is single issued.

Table 6-30 Multiply Accumulate Instruction Timing

Instruction	Result Latency TC16P-TC16E		Repeat Rate TC16P-TC16E		Instruction	Result Latency TC16-TC16E		Repeat Rate TC16P-TC16E	
MADD	3	2	1	1	**MSUB**	3	2	1	1
MADD.U	3	2	1	1	**MSUB.U**	3	2	1	1
MADDS	3	2	1	1	**MSUBS**	3	2	1	1
MADDS.U	3	2	1	1	**MSUBS.U**	3	2	1	1
MADD.H	3	2	1	1	**MSUB.H**	3	2	1	1
MADD.Q	3	2	1	1	**MSUB.Q**	3	2	1	1
MADDM.H	3	2	1	1	**MSUBM.H**	3	2	1	1
MADDMS.H	3	2	1	1	**MSUBMS.H**	3	2	1	1
MADDR.H	3	2	1	1	**MSUBR.H**	3	2	1	1
MADDR.Q	3	2	1	1	**MSUBR.Q**	3	2	1	1
MADDRS.H	3	2	1	1	**MSUBRS.H**	3	2	1	1
MADDRS.Q	3	2	1	1	**MSUBRS.Q**	3	2	1	1
MADDS.H	3	2	1	1	**MSUBS.H**	3	2	1	1
MADDS.Q	3	2	1	1	**MSUBS.Q**	3	2	1	1
MADDSU.H	3	2	1	1	**MSUBAD.H**	3	2	1	1
MADDSUM.H	3	2	1	1	**MSUBADM.H**	3	2	1	1
MADDSUMS.H	3	2	1	1	**MSUBADMS.H**	3	2	1	1
MADDSUR.H	3	2	1	1	**MSUBADR.H**	3	2	1	1
MADDSURS.H	3	2	1	1	**MSUBADRS.H**	3	2	1	1
MADDSUS.H	3	2	1	1	**MSUBADS.H**	3	2	1	1

For All MADD, MSUB and MUL type instructions the result latency is reduced to 1 for accumulator forwarding between similar instructions.

For MADD.Q, MADDS.Q, MSUB.Q, MSUBS.Q Instructions:

MADD.Q, MADDS.Q, MSUB.Q, MSUBS.Q	Result Latency TC1.6P	Result Latency TC1.6E	Repeat Rate TC1.6P	Repeat Rate TC1.6E
16 × 16	3	2	1	1
16 × 32	3	2	1	1
32 × 32	3	2	1	1

6.9.1.4 Control Flow Instruction Timing TC1.6P

Control flow instruction timing for TC1.6P is complicated by the use of branch target buffers and fetch FIFOs.

- Incorrectly predicted LS instructions incur a three cycle branch recovery penalty.
- Incorrectly predicted IP instructions incur a four cycle branch recovery penalty.
- Correctly predicted not taken branches incur no penalty
- Correctly predicted taken branch incur a penalty of up to two cycles depending on the state of the fetch FIFOs and the branch target buffer.
- Loop instructions incur the same penalty as an LS conditional jump instruction.

Assumptions

- All target locations yield a full instruction in one access (i.e. not 16-bits of a 32-bit instruction).
- All code fetches take a single cycle.
- Timing is best case; no cache misses for context operations, no pending stores.

Table 6-31 Control flow timing(TC1.6P)

Prediction-Result	Flow Latency (LS, LP)	Repeat rate (LS,LP)	Flow Latency (IP)	Repeat Rate (IP)
Correct Not-Taken	1	1	1	1
Correct Taken	1-2	1-2	1-2	1-2
Incorrect Not-Taken	3	4	3	4
Incorrect Taken	3	4	3	4

6.9.1.5 Control Flow Instruction Timing TC1.6E

The TC1.6E implements a simple static branch prediction scheme. 16 bit instruction format (either direction) and 32 bit format backwards conditional branches are predicted taken. 32 bit format forwards conditional branches are predicted not taken.

- Correctly predicted not taken branches incur no penalty.
- Correctly predicted taken branches incur a single cycle penalty.
- Incorrectly predicted branches incur a two cycle branch recovery penalty.
- Loop instructions incur the same penalty as LS conditional jump instructions.

Assumptions

- All target locations yield a full instruction in one access (i.e. not 16-bits of a 32-bit instruction).
- All code fetches take a single cycle.
- Timing is best case; no cache misses for context operations, no pending stores.

Table 6-32 Control flow timing(TC1.6E)

Prediction-Result	Flow Latency	Repeat rate
Correctly predicted, not taken	1	1
Correctly predicted, taken	2	2
Incorrectly predicted	3	3

6.9.2 Load-Store Pipeline Instructions

This section summarizes the Load-Store Pipeline instructions.

6.9.2.1 Address Arithmetic Timing

Each instruction is single issued.

Table 6-33 Address Arithmetic Instruction Timing

Instruction	Result Latency TC16P- TC16E		Repeat Rate TC16P- TC16E		Instruction	Result Latency TC16P -TC16E		Repeat Rate TC16P-TC16E	
Load Store Arithmetic Instructions									
ADD.A	1	1	1	1	**GE.A**	1	1	1	1
ADDIH.A	1	1	1	1	**LT.A**	1	1	1	1
ADDSC.A	2	1	1	1	**NE.A**	1	1	1	1
ADDSC.AT	2	1	1	1	**NEZ.A**	1	1	1	1
EQ.A	1	1	1	1	**SUB.A**	1	1	1	1
EQZ.A	1	1	1	1	**NOP**	1	1	1	1
Trap and Interrupt Instructions									
DEBUG	–	–	1	1	**TRAPSV**[1)]	–	–	1	1
DISABLE	–	–	1	1	**TRAPV**[1)]	–	–	1	1
ENABLE	–	–	1	1	**RSTV**	–	–	1	1
RESTORE	–	–	1	1	**WAIT**[2)]	-	-	1	1
Move Instructions									
MFCR	2	2	1	1	**MOV.A**	1	1	1	1
MTCR	1	1	1	1	**MOV.AA**	1	1	1	1
MOVH.A	1	1	1	1	**MOV.D**	1	1	1	1
Sync Instructions									
DSYNC	–	–	1	1	**ISYNC**[3)]	–	–	1	1

1) Execution cycles when no TRAP is taken. The execution timing in the case of raising these TRAPs is the same as other TRAPs such as SYSCALL.

2) The latency of the WAIT instruction is hidden by the context save of the interrupt. Effective latency is zero.

3) Repeat rate assumes that code refetch takes a single cycle.

6.9.2.2 CSA Control Flow Instruction Timing

This section summarizes the timing of CSA Control Flow instructions.

- All targets yield a full instruction in one access (not 16-bits of a 32-bit instruction).
- All code fetches take a single cycle. Timing is best case; no cache misses for context operations, no pending stores.

Table 6-34 CSA Control Flow Instruction Timing

Instruction	Result Latency TC16P-TC16E		Repeat Rate TC16P-TC16E		Instruction	Result Latency TC16P-TC16E		Repeat Rate TC16P-TC16E	
CALL	4-8	4	4-8	4	**SYSCALL**	4-8	4	4-8	4
CALLA	4-8	4	4-8	4	**SVLCX**	4-8	4	4-8	4
CALLI	4-8	4	4-8	4	**RSLCX**	4-8	4	4-8	4
RET	4-8	4	4-8	4	**RFE**	4-8	4	4-8	4
BISR	4-8	4	4-8	4	**RFM**	4-8	4	4-8	4
FCALL	1	1	1	1	**FCALLA**	1	1	1	1
FCALLI	1	1	1	1	**FRET**	1	1	1	1

Access to DSPR require 4 cycles, accesses to cached external memory require 8 cycles (TC1.6P only).

6.9.2.3 Load Instruction Timing

Load instructions can produce two results if they use the pre-increment, post-increment, circular or bit-reverse addressing modes. Hence, in those cases there are two latencies that must be specified, the result latency for the value loaded from memory and the address latency for using the updated address register result.

- Each instruction is single issued.
- The memory references is naturally aligned.
- The memory accessed takes a single cycle to return a data item.
- Timing is best case; no cache misses, no pending stores.

Table 6-35 Load Instruction Timing TC1.6P

Instruction	Address Latency	Result Latency	Repeat Rate	Instruction	Address Latency	Result Latency	Repeat Rate
Load Instructions							
LD.A	1	3	1	**LD.Q**	1	2	1
LD.B	1	2	1	**LD.W**	1	2	1

Table 6-35 Load Instruction Timing TC1.6P (cont'd)

Instruction	Address Latency	Result Latency	Repeat Rate	Instruction	Address Latency	Result Latency	Repeat Rate
LD.BU	1	2	1	**LDLCX**	5-9	5-9	5
LD.D	1	2	1	**LDUCX**	5-9	5-9	5
LD.DA	1	3	1	**SWAP.W**	2	3	2
LD.H	1	2	1	**LEA**[1)]	–	1	1
LD.HU	1	2	1	**CMPSWAP**	2	3	1
SWAPMSK	2	3	1				

1) The addressing mode returning an updated address is not relevant for this instruction.

Table 6-36 Load Instruction Timing TC1.6E

Instruction	Address Latency	Result Latency	Repeat Rate	Instruction	Address Latency	Result Latency	Repeat Rate
Load Instructions							
LD.A	1	3	1	**LD.Q**	1	2	1
LD.B	1	2	1	**LD.W**	1	2	1
LD.BU	1	2	1	**LDLCX**	5	5	5
LD.D	1	2	1	**LDUCX**	5	5	5
LD.DA	1	3	1	**SWAP.W**	2	3	2
LD.H	1	2	1	**LEA**[1)]	–	1	1
LD.HU	1	2	1	**CMPSWAP**	2	3	1
SWAPMSK	2	3	1				

1) The addressing mode returning an updated address is not relevant for this instruction.

6.9.2.4 Store Instruction Timing

Cache and Store instructions similar to Load instructions will have a result for the pre-increment, post-increment, circular or bit-reverse addressing modes, but do not produce a ‘memory’ result.

- Each instruction is single issued.
- The memory references is naturally aligned.
- The memory accessed takes a single cycle to accept a data item.
- Timing is best case; no cache misses, no pending stores.

Table 6-37 Cache and Store Instruction Timing

Instruction	Result Latency TC16P- TC16E		Repeat Rate TC16P- TC16E		Instruction	Result Latency TC16P -TC16E		Repeat Rate TC16P-TC16E	
Cache Instructions									
CACHEA.I	1	1	1	1	**CACHEA.WI**[1]	1	1	1	1
CACHEA.W[1]	1	1	1	1	**CACHEI.W**	1	1	1	1
CACHEI.WI	1	1	1	1	**CACHEI.I**	1	1	1	1
Store Instructions									
ST.A	1	1	1	1	**ST.T**	1	1	2	2
ST.B	1	1	1	1	**ST.W**	1	1	1	1
ST.D	1	1	1	1	**STLCX**	1	1	9	5
ST.DA	1	1	1	1	**STUCX**	1	1	9	5
ST.H	1	1	1	1	**LDMST**	1	1	2	2
ST.Q	1	1	1	1					

1) Repeat rate assumes that no memory writeback operation occurs. Otherwise the repeat rate will depend upon the time for the castout buffers to clear.

6.9.3 Floating Point Pipeline Timing

These instructions are only valid if the optional Floating Point Unit is implemented.

Each instruction is single issued.

Table 6-38 Floating Point Instruction Timing

Instruction	Result Latency TC16P- TC16E		Repeat Rate TC16P- TC16E		Instruction	Result Latency TC16P -TC16E		Repeat Rate TC16P-TC16E	
Floating Point Instructions									
ADDF	2	2	1	1	**ITOF**	2	1	1	1
CMP.F	1	1	1	1	**MADD.F**	3	2	1	1
DIV.F	8	7	6	6	**MSUB.F**	3	2	1	1
FTOI	2	1	1	1	**MUL.F**	2	2	1	1
FTOIZ	2	1	1	1	**Q31TOF**	2	1	1	1
FTOQ31	2	1	1	1	**QSEED.F**	1	1	1	1
FTOQ31Z	2	1	1	1	**SUB.F**	2	2	1	1
FTOU	2	1	1	1	**UPDFL**	–	–	1	1
FTOUZ	2	1	1	1	**UTOF**	2	1	1	1

6.10 Program Memory Interface (PMI)

The program Memory Interface (PMI) provides the instruction stream to the CPU.

6.10.1 TC1.6P PMI Description

Figure 6-9 shows the block diagram of the Program Memory Interface (PMI) of the TC1.6P.

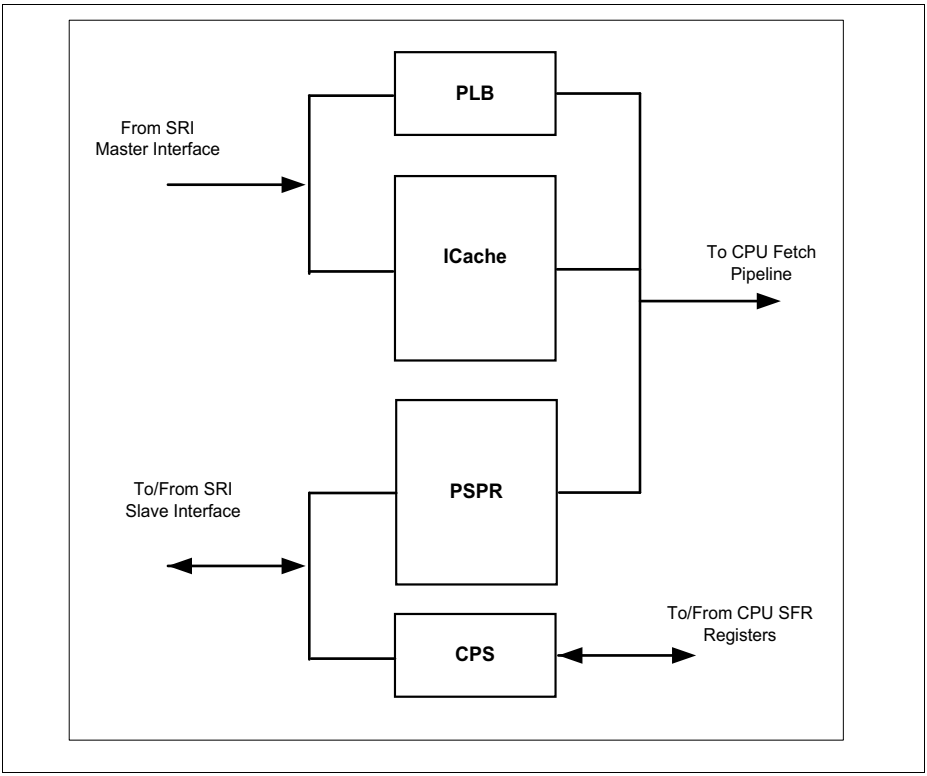


Figure 6-9 PMI Block Diagram

The Program Memory Interface (PMI) has the following features:

- Program Cache (PCACHE)
  - Two-way set associative cache
  - LRU (Least-Recently Used) replacement algorithm
  - Cache line size: 256 bits (4 double-words)
  - Validity granularity: One valid bit per cache line
  - Single cycle access for hit.

  - PCACHE can be globally invalidated to provide support for software cache coherency (to be handled by the programmer)
  - PCACHE can be bypassed to provide a direct fetch from the CPU to on-chip and off-chip resources
  - PCACHE refill mechanism:
    Critical word first, line wrap around, streaming to CPU
- Program Scratchpad memory (PSPR)
- CPU interface
  - Supporting 64bit aligned fetches. (TC1.6P)
- CPU Slave interface (CPS)
- Shared Resource Interconnect Bus (SRI) Master Interface
- Shared Resource Interconnect Bus (SRI) Slave Interface to scratchpad RAM and CPS.
- All PMI SRAMs (PSPR, PCACHE, and cache tag SRAM) are ECC protected
  - ECC is calculated on 64bit data for the PSPR and PCACHE

6.10.1.1 TC1.6P Scratchpad RAM

The TC1.6P of Program scratchpad RAM provides a fast, deterministic program fetch access from the CPU for use by performance critical code sequences.

- CPU program fetch accesses to scratchpad RAM are never cached in the program cache and are always directly targeted to the scratchpad RAM.

The TC1.6P CPU fetch interface will generate aligned accesses (64-bit), which will result in 64-bits of instruction being returned to the CPU.

Note that the CPU Fetch Unit can only read from the scratchpad RAM and can never write to it.

The scratchpad RAM may also be accessed from the SRI Slave interface by another bus master, such as the Data Memory Interface (DMI). The scratchpad RAM may be both read and written from the SRI. The SRI Slave interface supports all SRI transaction types.

The scratchpad RAM is ECC protected across 64bit data.

6.10.1.2 TC1.6P Program Cache

The program Cache is a two-way set-associative cache with a Least-Recently-Used (LRU) replacement algorithm. Each PCACHE line contains 256 bits of instruction and associated ECC bits.

CPU program fetch accesses which target a cacheable memory segment (and where the PCACHE is not bypassed) target the PCACHE. If the requested address and its associated instruction are found in the cache (Cache Hit), the instruction is passed to the CPU Fetch Unit without incurring any wait states. If the address is not found in the cache (Cache Miss), the PMI cache controller issues a cache refill sequence and wait states

are incurred whilst the cache line is refilled. The fetch request always returns an aligned packet to the CPU.

The program cache is ECC protected on 64bit data.

The program TAG Ram is ECC protected on 22bit data.

Errors detected at the ECC decoders are signaled to the EMM via the MBIST logic of the associated SRAM

Program Cache Refill Sequence

Program Cache refills are performed using a critical double-word first strategy with cache line wrapping such that the refill size is always 4 double-words. PCACHE refills are always performed in 64-bit quantities. A refill sequence will always affect only one cache line. There is no prefetching of the next cache line.

PCACHE refills are therefore implemented using an SRI Burst Transfer 4 (BTR4) transfers. The program cache supports instruction streaming, meaning that it can deliver available instruction half-words to the CPU Fetch Unit whilst the refill operation is ongoing.

Program Cache Bypass

The Program Cache may be bypassed, under control of PCON0.PCBYP, to provide a direct instruction fetch path for the CPU Fetch Unit. The default value of PCON0.PCBYP is such that the PCACHE is bypassed after reset.

Whilst PCACHE bypass is enabled, a fetch request by the CPU to a cacheable address will result in a forced cache miss, such that the cache controller issues a standard refill sequence and supplies instruction half-words to the CPU using instruction streaming, without updating the cache contents. Any valid cache lines within the PCACHE will remain valid and unchanged whilst the PCACHE is bypassed. As such, instruction fetch requests to cacheable addresses with PCACHE bypass enabled behave identically to instruction fetch requests to non-cacheable addresses.

The PCON0 register is endinit protected.

Program Cache Invalidation

The PMI does not have automatic cache coherency support. Changes to the contents of memory areas external to the PMI that may have already been cached in the PCACHE are not detected. Software must provide the cache coherency in such a case. The PMI supports this via the cache invalidation function. The PCACHE contents may be globally invalidated by writing a ‘1’ to PCON1.PCINV. The PCACHE invalidation is performed over 64 cycles by a hardware state machine which cycles through the PCACHE entries marking each as invalid. During an invalidate sequence the CPU may continue to fetch instructions from non-cacheable memory. Any attempt to fetch instructions from a cacheable memory location during an invalidation sequence will result in the CPU

stalling until the sequence completes. The status of the PCACHE invalidation sequence may be determined by reading the PCON1.PCINV bit.

6.10.1.3 TC1.6P Program Line Buffer

The PMI module contains a 256-bit Program Line Buffer (PLB). Program fetch requests to non-cacheable addresses (or to cacheable addresses with the cache in bypass) utilize the PLB as a single line cache. A single valid bit is associated with the PLB, denoting that the PLB contents are valid. As such all fetch requests resulting in an update of the PLB, whether to a cacheable address or not, are implemented as SRI Burst Transfer 4 (BTR4) transactions, with the critical double-word of the PLB line being fetched first size. The PLB may be invalidated by writing PCON1.PBINV.

6.10.1.4 TC1.6P CPU Slave Interface (CPS)

The CPU Slave Interface provides access from the SRI bus to the CPU CSFR and SFR registers.

6.10.2 TC1.6E PMI Description

Figure 6-10 shows the block diagram of the Program Memory Interface (PMI) of the TC1.6E.

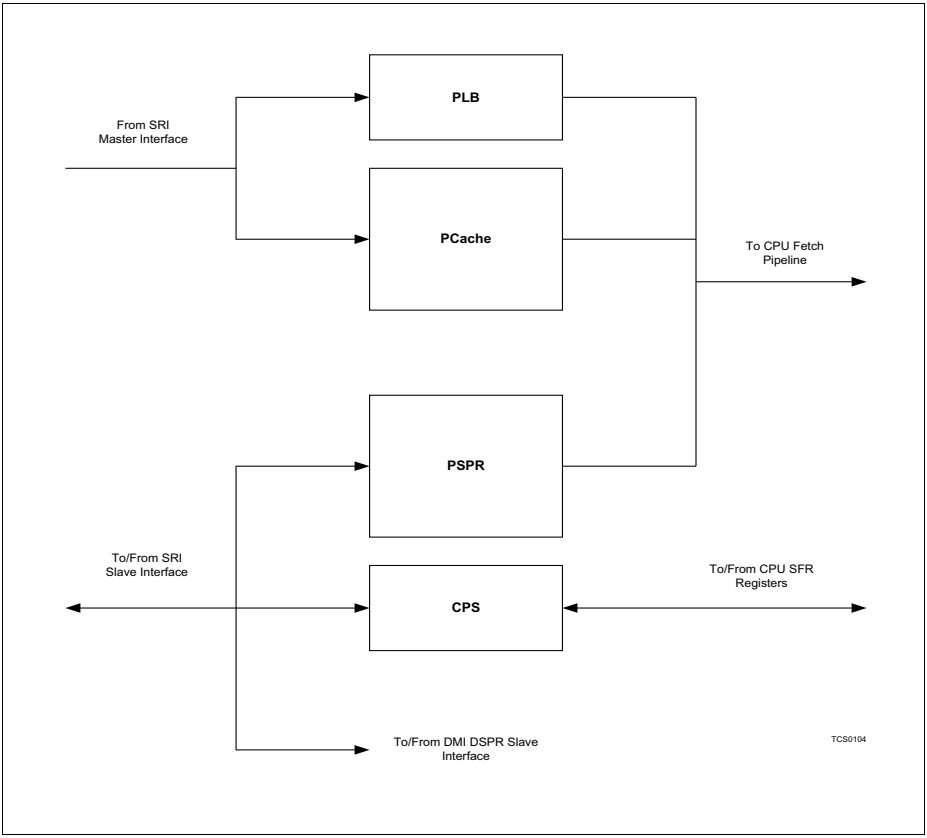


Figure 6-10 PMI Block Diagram

The Program Memory Interface (PMI) has the following features:

- Program Cache (PCACHE)
  - Two-way set associative cache
  - LRU (Least-Recently Used) replacement algorithm
  - Cache line size: 256 bits (4 double-words)
  - Validity granularity: One valid bit per cache line

  - PCACHE can be globally invalidated to provide support for software cache coherency (to be handled by the programmer)
  - PCACHE can be bypassed to provide a direct fetch from the CPU to on-chip and off-chip resources
  - PCACHE refill mechanism:
    Critical word first, line wrap around, streaming to CPU
- Program Scratchpad memory (PSPR)
- CPU interface
  - Supporting 32-bit aligned fetches. (TC1.6E)
- CPU Slave interface (CPS)
- Shared Resource Interconnect Bus (SRI) Master Interface
- Shared Resource Interconnect Bus (SRI) Slave Interface to Program Scratchpad RAM and CPU Slave interface.
  - SRI Slave Interface is shared with DMI for Data Scratchpad RAM access.
- All PMI SRAMs (PSPR, PCACHE, and cache tag SRAM) are ECC protected.
  - ECC is calculated on 32bit data for the PSPR and PCACHE

6.10.2.1 TC1.6E Program Scratchpad RAM

The Program Scratchpad RAM provides a fast, deterministic program fetch access from the CPU for use by performance critical code sequences.

- CPU program fetch accesses to scratchpad RAM are never cached in the program cache and are always directly targeted to the scratchpad RAM.

The TC1.6E CPU fetch interface will generate aligned accesses (32-bit), which will result in 32-bits of instruction being returned to the CPU.

Note that the CPU Fetch Unit can only read from the scratchpad RAM and can never write to it.

The scratchpad RAM may also be accessed from the SRI Slave interface by another bus master, such as the Data Memory Interface (DMI). The scratchpad RAM may be both read and written from the SRI. The SRI Slave interface supports all SRI transaction types.

The scratchpad RAM is ECC protected across 32bit data.

Errors detected at the ECC decoders are signaled to the EMM via the MBIST logic of the associated SRAM

6.10.2.2 TC1.6E Program Cache

The Program Cache is a two-way set-associative cache with a Least-Recently-Used (LRU) replacement algorithm. Each PCACHE line contains 256 bits of instruction along with associated ECC bits.

CPU program fetch accesses which target a cacheable memory segment (and where the PCACHE is not bypassed) target the PCACHE. If the requested address and its

associated instruction are found in the cache (Cache Hit), the instruction is passed to the CPU Fetch Unit without incurring any wait states. If the address is not found in the cache (Cache Miss), the PMI cache controller issues a cache refill sequence and wait states are incurred whilst the cache line is refilled. The fetch request always returns an aligned packet to the CPU.

The Program cache is ECC protected on 32bit data.

The Program TAG Ram is ECC protected on 22bit data.

Errors detected at the ECC decoders are signaled to the EMM via the MBIST logic of the associated SRAM

Program Cache Refill Sequence

Program Cache refills are performed using a critical double-word first strategy with cache line wrapping such that the refill size is always 4 double-words. PCACHE refills are always performed in 64-bit quantities. A refill sequence will always affect only one cache line. There is no prefetching of the next cache line.

PCACHE refills are therefore implemented using an SRI Burst Transfer 4 (BTR4) transfers. The program cache supports instruction streaming, meaning that it can deliver available instruction half-words to the CPU Fetch Unit whilst the refill operation is ongoing.

Program Cache Bypass

The Program Cache may be bypassed, under control of PCON0.PCBYP, to provide a direct instruction fetch path for the CPU Fetch Unit. The default value of PCON0.PCBYP is such that the PCACHE is bypassed after reset.

Whilst PCACHE bypass is enabled, a fetch request by the CPU to a cacheable address will result in a forced cache miss, such that the cache controller issues a standard refill sequence and supplies instruction half-words to the CPU using instruction streaming, without updating the cache contents. Any valid cache lines within the PCACHE will remain valid and unchanged whilst the PCACHE is bypassed. As such, instruction fetch requests to cacheable addresses with PCACHE bypass enabled behave identically to instruction fetch requests to non-cacheable addresses.

The PCON0 register is endinit protected.

Program Cache Invalidation

The PMI does not have automatic cache coherency support. Changes to the contents of memory areas external to the PMI that may have already been cached in the PCACHE are not detected. Software must provide the cache coherency in such a case. The PMI supports this via the cache invalidation function. The PCACHE contents may be globally invalidated by writing a ‘1’ to PCON1.PCINV. The PCACHE invalidation is performed over 64 cycles by a hardware state machine which cycles through the PCACHE entries

marking each as invalid. During an invalidate sequence the CPU may continue to fetch instructions from non-cacheable memory. Any attempt to fetch instructions from a cacheable memory location during an invalidation sequence will result in the CPU stalling until the sequence completes. The status of the PCACHE invalidation sequence may be determined by reading the PCON1.PCINV bit.

6.10.2.3 TC1.6E Program Line Buffer

The PMI module contains a 256-bit Program Line Buffer (PLB). Program fetch requests to non-cacheable addresses (or to cacheable addresses with the cache in bypass) utilize the PLB as a single line cache. A single valid bit is associated with the PLB, denoting that the PLB contents are valid. As such all fetch requests resulting in an update of the PLB, whether to a cacheable address or not, are implemented as SRI Burst Transfer 4 (BTR4) transactions, with the critical double-word of the PLB line being fetched first size. The PLB may be invalidated by writing PCON1.PBINV.

6.10.2.4 TC1.6E CPU Slave Interface (CPS)

The CPU Slave Interface provides access from the SRI bus to the CPU CSFR and SFR registers.

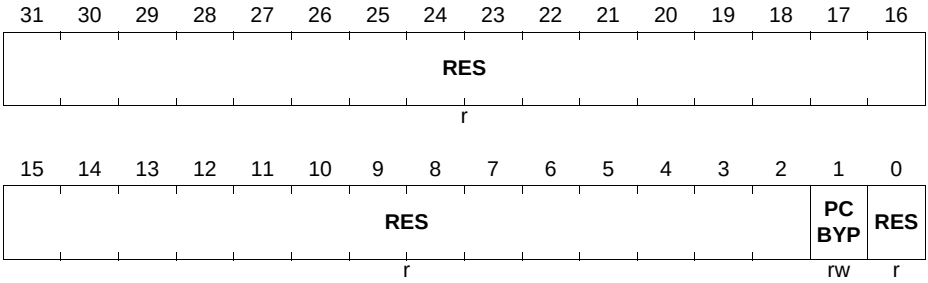
6.10.3 PMI Registers

Three control registers are control the operation of the Program Memory Interface. These registers and their bits are described in this section.

6.10.3.1 PMI Register Descriptions

PMI Control Register 0

PCON0
Program Memory Control Register 0
(CSFR_Base + 920C$_H$) **Reset Value: 0000 0002$_H$**



Field	Bits	Type	Description
RES	0	r	**Reserved** Read as 0; should be written with 0.
PCBYP	1	rw	**Program Cache Bypass** 0_B Cache enabled 1_B Cache bypassed (disabled)
RES	[31:2]	r	**Reserved** Read as 0; should be written with 0.

Note: This register is endinit protected.

PMI Control Register 1

PCON1
Program Memory Control Register 1

(CSFR_Base + 9204_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES															
r															

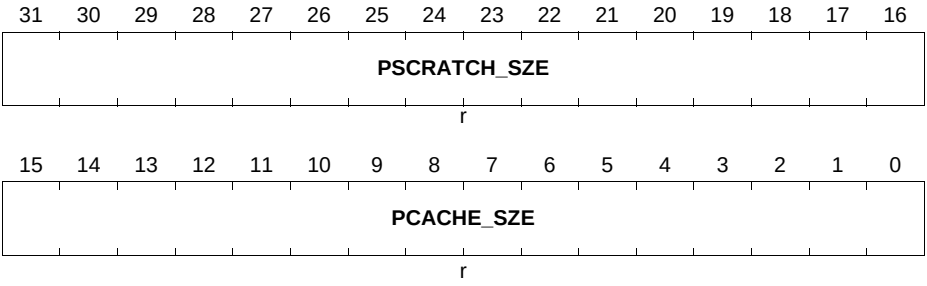
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES														PB INV	PC INV
r														rw	rwh

Field	Bits	Type	Description
PCINV	0	rwh	**Program Cache Invalidate** Write Operation: 0_B No effect. Normal program cache operation. 1_B Initiate invalidation of entire program cache. Read Operation: 0_B Normal operation. Program cache available. 1_B Program cache invalidation in progress. Program cache unavailable.
PBINV	1	rw	**Program Buffer Invalidate** Write Operation: 0_B No effect. Normal program line buffer operation. 1_B Invalidate the program line buffer. This field returns 0 when read.
RES	[31:2]	r	**Reserved** Read as 0; should be written with 0.

PMI Control Register 2

PCON2 contains size information for the Program memory system.

PCON2
Program Memory Control Register 2
(CSFR_Base + 9208_H) Reset Value: CPU Dependent



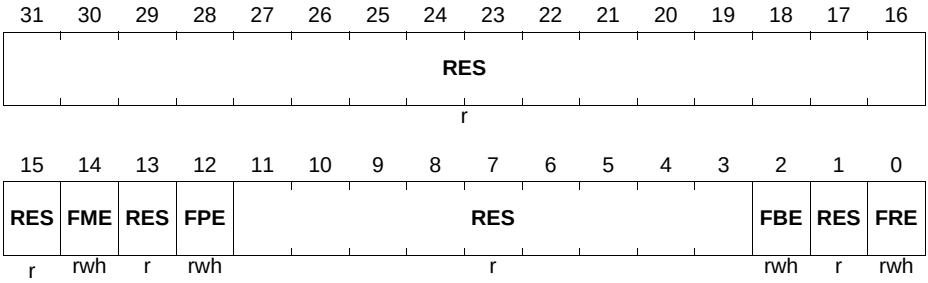
Field	Bits	Type	Description
PCACHE_SZE	[15:0]	r	**Program Cache (PCACHE) Size** In KBytes
PSCRATCH_SZE	[31:16]	r	**Program Scratch Size** In KBytes

Program Memory Interface Synchronous Trap Register (PSTR)

PSTR contains synchronous trap information for the program memory system. The register is updated with trap information for PSE traps to aid the localisation of faults.

The register is only set whenever a trap is detected and the register has no bits already set. It is cleared by a CSFR write (independent of data value).

PSTR
Program Synchronous Trap Register
(CSFR_Base + 9200_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
FRE	0	rwh	**Fetch Range Error** Code fetch from local PSPR area outside of physically implemented memory
RES	1	r	**Reserved**
FBE	2	rwh	**Fetch Bus Error** Code fetch from bus address giving an error
RES	[11:3]	r	**Reserved**
FPE	12	rwh	**Fetch Peripheral Error** Code fetch from peripheral space
RES	13	r	**Reserved**
FME	14	rwh	**Fetch MSIST Error** Code fetch from memory mapped PTAG.
RES	[31:15]	r	**Reserved**

Fetch Range Error

A Fetch Range Error occurs whenever an access to the Program Scratch is outside the range of the SRAM.

Fetch Global Address Error

A Fetch Global Address Error occurs whenever an there is access to an address that cannot be translated to a valid global address.This will occur for accesses in the range D1000000 to D7FFFFFF

Fetch Bus Error

A Fetch bus error will be set whenever the SRI flags an error due a fetch from external memory. This will be set for both direct fetches from the bus and for cache refills.

Fetch Peripheral Error

A Fetch peripheral error will be flagged whenever a fetch is attempted to peripheral space.

Fetch MSIST Error

During SIST mode, a fetch from the PTAG will cause a PSE trap to occur.

6.11 Data Memory Interface (DMI)

The Data Memory Interface (DMI) provides data values to the CPU and stores data values supplied by the CPU.

6.11.1 TC1.6P DMI Description

This figure shows the block diagram of the Data Memory Interface (DMI) of the TC1.6P.

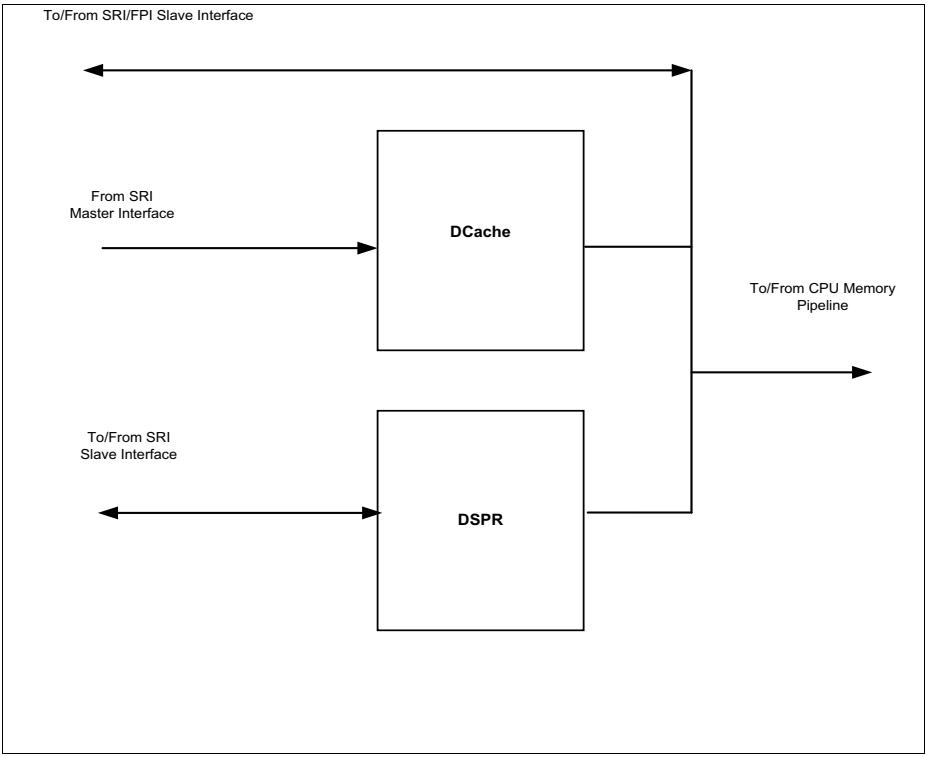


Figure 6-11 DMI Block Diagram

The Data Memory Interface (DMI) has the following features:

- Data Scratchpad Ram (DSPR)
  - Supporting unaligned access (16-bit aligned) with no penalty.
- Data Memory (DCACHE):
  - Two-way set associative cache, Least recently used (LRU) replacement algorithm
  - Cache line size: 256 bits
  - Validity granularity: One valid bit per cache line

  - Write-back Cache: Writeback granularity: 256 bits
  - Refill mechanism: full cache line refill
  - Single cycle access for hit.
- CPU interface
- Shared Resource Interconnect Bus (SRI) Master interface
- Shared Resource Interconnect (SRI) Slave interface to DSPR
- Flexible Peripheral Interconnect Bus (FPI) Master interface for fast access to peripherals
- All DMI SRAMs (DSPR, DCACHE, and cache tag SRAM) are ECC protected

6.11.1.1 TC1.6P Data Scratchpad RAM (DSPR)

The DSPR provides fast, deterministic data access to the CPU for use by performance critical code sequences.

The DSPR is organised as multiple memory “towers”. This organisation allow the CPU to access 64bits of data from any 16bit aligned address

The DSPR may also be accessed from the SRI Slave interface by another bus master, with both read and write transactions supported. The DSPR may be accessed by the SRI Slave interface using any SRI transaction type, including burst transfers. In accordance with the SRI protocol, accesses to the SRI Slave interface must be naturally aligned.

All TC1.6P Data scratchpad RAMs are ECC protected.

Errors detected at the ECC decoders are signaled to the EMM via the MBIST logic of the associated SRAM

6.11.1.2 TC1.6P Data Cache (DCACHE)

The DCache is a Two-way set-associative cache with a Least-Recently-Used (LRU) replacement algorithm. Each line contains 256 bits of data along with ECC bits. A single valid bit and a single dirty bit are associated with each line.

CPU data accesses to a cacheable memory segment target the DCache. If the requested address and its associated data are found in the cache (Cache Hit), the data is passed to/from the CPU Load-Store Unit without incurring any wait states. If the address is not found in the cache (Cache Miss), the DMI cache controller issues a cache refill sequence and wait states are incurred whilst the cache line is refilled. The CPU load-store interface will generate unaligned accesses (16-bit aligned), which will result in up to 64-bits of data being transferred to or from the CPU (for non-context operations). If the data access is made within a DCache line, no matter the alignment, and a cache hit is detected then the requested data is returned to the CPU in a single cycle. If the data access is made to the end of a DCache line, such that the requested data would span two DCache lines, a single wait cycle is incurred (if both cache lines are present in the cache, otherwise a refill sequence is required for the missing cache line(s)).

The TC1.6P data cache is of the writeback type. When the CPU writes to a cacheable location the data is merged with the corresponding cache line and not written to main memory immediately. Associated with each cache line is a single ‘dirty’ bit, to denote that the data in the cache line has been modified. Whenever a CPU load-store access results in a cache miss, and each of the potential cache ways that could hold the requested cache line are valid, one of the cache lines is chosen for eviction based upon the LRU replacement algorithm. The line selected for eviction is then checked to determine if it has been modified using its dirty bit. If the line has not been modified the line is discarded and the refill sequence started immediately. If the line has been modified then the dirty data is first written back to main memory before the refill is initiated. Due to the single dirty bit per cache line, 256 bits of data will always be written back, resulting in a SRI Burst-4 Transfer (BTR4) transactions.

Data Cache refills always result in the full cache line being refilled, with the critical double-word of the DCache line being fetched first. A refill sequence will always affect only one cache line. There is no prefetching of the next cache line. Due to the uniform size of DCache refill sequences, such refills are always implemented using SRI Burst Transfer 4 (BTR4) transactions.

All TC1.6P Cache SRAMs are ECC protected.

All TC1.6P TAG SRAMs are ECC protected

Errors detected at the ECC decoders are signaled to the EMM via the MBIST logic of the associated SRAM

Data Cache Bypass

The Data Cache may be bypassed, under control of DCON0.DCBYP, to provide a direct data access path to memory. The default value of DCON0.DCBYP is such that the data cache is bypassed after reset.

Whilst the data cache bypass is enabled, a data access request by the CPU to a cacheable address will result in a forced cache miss. Any valid cache lines within the data cache will remain valid and unchanged whilst the data cache is bypassed. As such data accesses to cacheable addresses with the data cache bypass enabled behave identically to data accesses to non-cacheable addresses.

The DCON0 register is endinit protected.

6.11.2 TC1.6E DMI Description

This figure shows the block diagram of the Data Memory Interface (DMI) of the TC1.6E.

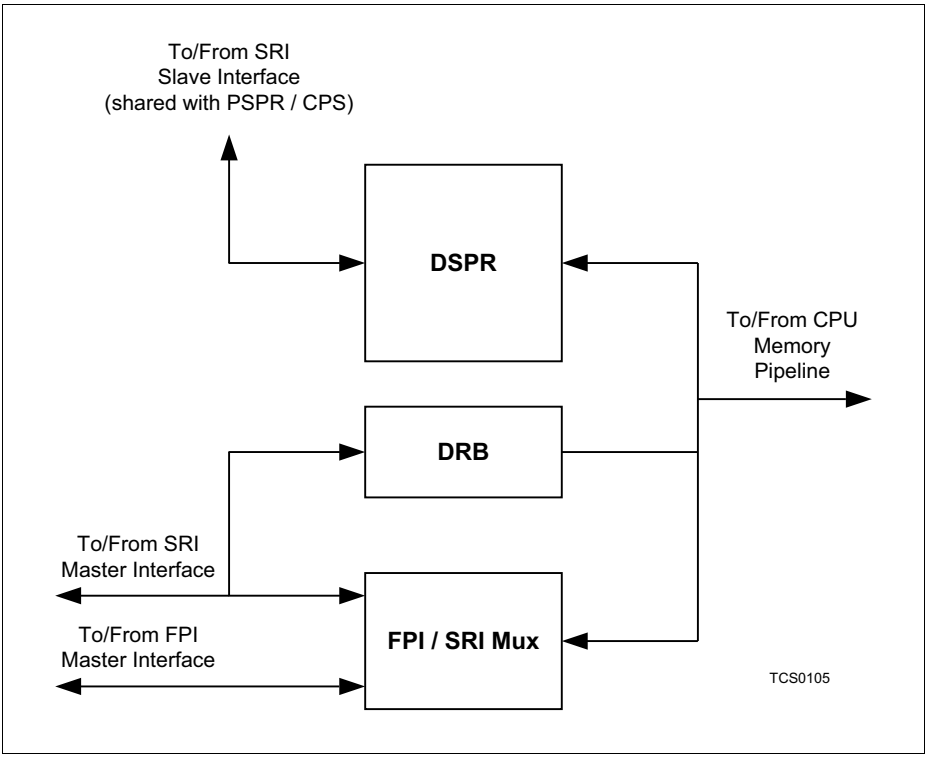


Figure 6-12 DMI Block Diagram

The Data Memory Interface (DMI) has the following features:

- Data Scratchpad Ram (DSPR)
  - Supporting unaligned access (16-bit aligned) with no penalty.
- Data Read Buffer (DRB)
  - Four-entry fully associative Data Read Buffer
  - Least recently used (LRU) replacement algorithm
  - Buffer line size: 256 bits
  - Validity granularity: One valid bit per buffer line
  - Refill mechanism: full buffer line refill, critical double-word first
- CPU interface
- Shared Resource Interconnect Bus (SRI) Master interface

- Flexible Peripheral Interconnect Bus (FPI) Master interface for fast access to peripherals
  - CPU accesses to lower half of segment F_H are directed to FPI master interface, other accesses directed to SRI master interface
- Shared Resource Interconnect (SRI) Slave interface to DSPR
  - SRI slave interface shared with PMI for access to PSPR / CPS
- All TC1.6E DMI SRAM (DSPR) ECC protected

6.11.2.1 TC1.6E Data Scratchpad RAM (DSPR)

The DSPR provides fast, deterministic data access to the CPU for use by performance critical code sequences.

The DSPR is organised as multiple memory “towers”. This organisation allow the CPU to access 64bits of data from any 16bit aligned address

The DSPR may also be accessed from the SRI Slave interface by another bus master, with both read and write transactions supported. The DSPR may be accessed by the SRI Slave interface using any SRI transaction type, including burst transfers. In accordance with the SRI protocol, accesses to the SRI Slave interface must be naturally aligned.

Errors detected at the ECC decoders are signaled to the EMM via the MBIST logic of the associated SRAM

6.11.2.2 TC1.6E Data Read Buffer (DRB)

The TC1.6E implements a 128-byte Data Read Buffer (DRB). The DRB contains four fully associative buffer lines, with a Least-Recently-Used (LRU) replacement algorithm. Each buffer line contains 256-bits of data. A single valid bit is associated with each buffer line. The DRB will buffer CPU read data from cacheable memory segments.

CPU load data accesses, where the addressed memory segment is defined as cacheable, target the DRB. If the requested address and its associated data are found in the DRB (Buffer Hit), the data is passed to/from the CPU Load-Store Unit without incurring any wait states. If the address is not found in the DRB (Buffer Miss), the DMI controller issues a buffer refill sequence and wait states are incurred whilst the buffer line is refilled. The CPU load-store interface will generate unaligned accesses (16-bit aligned), which will result in up to 64-bits of data being transferred to or from the CPU (for non-context operations). If the data access is made within a DRB buffer line, no matter the alignment, and a buffer hit is detected then the requested data is returned to the CPU in a single cycle. If the data access is made to the end of a DRB line, such that the requested data would span two DRB lines, a single wait cycle is incurred (if both buffer lines are present in the DRB, otherwise a refill sequence is required for the missing buffer line(s)).

DRB refills always result in the full buffer line being refilled, with the critical double-word of the DRB line being fetched first. A refill sequence will always affect only one buffer

line. There is no prefetching of the next buffer line. Due to the uniform size of DRB refill sequences, such refills are always implemented using SRI Burst Transfer 4 (BTR4) transactions.

The TC1.6E DRB will buffer read data only. Any CPU store access which results in a hit in a DRB line will invalidate that DRB line and the store will continue as normal, bypassing the DRB. In addition, the execution of any CACHEA.xx or CACHEI.xx instruction by a TC1.6E CPU will invalidate all its DRB lines.

DRB Bypass

The DRB may be bypassed, under control of DCON0.DCBYP, to provide a direct data access path to memory. The default value of DCON0.DCBYP is such that the DRB is bypassed after reset.

Whilst the DRB bypass is enabled, a data read request by the CPU to a cacheable address will result in a forced buffer miss. Any valid lines within the DRB will remain valid and unchanged whilst the DRB is bypassed. As such data reads to cacheable addresses with the DRB bypass enabled behave identically to data accesses to non-cacheable addresses.

The DCON0 register is endinit protected.

6.11.3 DMI Trap Generation

CPU data accesses to the DMI may encounter one of a number of potential error conditions, which result in one of the following trap conditions being reported by the DMI.

ALN Trap

An ALN trap is raised for the following conditions:

- An access whose effective address does not conform to the alignment rules
- An access where the length, size or index of a circular buffer is incorrect

Whenever an ALN trap occurs, the DSTR (Data Synchronous Trap Register) and the DEADD (Data Error Address Register) CSFRs are updated.

MEM Trap

A MEM trap is raised for the following conditions:

- An access whose effective address has a different segment to that of the base address (Segment Difference Error)
- An access whose effective address causes the data to span two segments (Segment Crossing Error)
- A memory address is used to access a CSFR area (CSFR Access Error)
- A context load or store instruction is executed where the effective address is not within the local DSPR range (Context Location Error)(TC1.6E only)

Whenever a MEM trap occurs, the DSTR (Data Synchronous Trap Register) and the DEADD (Data Error Address Register) CSFRs are updated.

DSE Trap

A DSE trap is raised for the following conditions:

- An access outside the range of the DSPR (Scratch Range Error)
- An access to the lower half of segment C which cannot be translated into a global address, i.e. from C1000000 to C7FFFFFF (Global Address Error)
- An error on the bus for an external accesses due to a load (Load Bus Error)
- An error from the bus during a cache refill (Cache Refill Error)
- An error during a load whilst in SIST mode (Load MSIST Error)
- An error generated by the overlay system during a load.

Whenever a DSE trap occurs, the DSTR (Data Synchronous Trap Register) and the DEADD (Data Error Address Register) CSFRs are updated.

DAE Trap

A DAE trap is raised for the following conditions:

- An error on the bus for an external accesses due to a store (Store Bus Error)
- An error on the bus due to a cache writeback (Cache writeback Error)(TC1.6P only)
- An error from the bus due to a cache flush (Cache Flush Error)(TC1.6P only)
- An error due to a store whilst in SIST mode (Store MSIST Error)
- An error generated by the overlay system during a store.

Whenever a non-inhibited DAE trap occurs, the DATR (Data Asynchronous Trap Register) and the DEADD (Data Error Address Register) CSFRs are updated. DAE traps are inhibited if the DATR register is non-zero.

Data Memory Protection Traps

Data memory protection traps (MPW, MPR, MPP, MPN) are raised by the memory protection system when a protection violation occurs. Whenever a data memory protection trap occurs the DSTR (Data synchronous trap register) and the DEADD (Data Error Address Register) are updated.

6.11.4 DMI Registers

Two control registers and three trap flag registers control the operation of the DMI. These registers and their bits are described in this section.

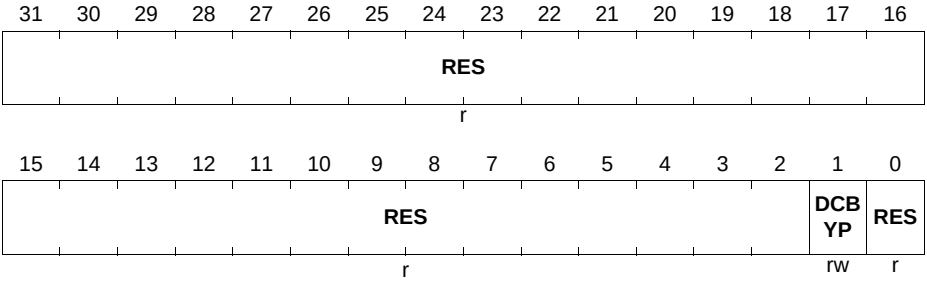
6.11.4.1 DMI Register Descriptions

Note: There is no DCON1 register in this implementation.

Data Memory Control Register 0 (DCON0)

The DCON0 register allows the Data cache to be bypassed.

DCON0
Data Memory Control Register
(CSFR_Base + 9040_H) **Reset Value: $0000\ 0002_H$**



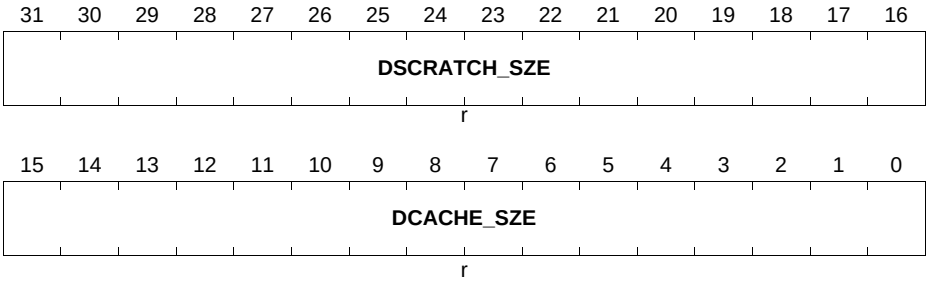
Field	Bits	Type	Description
RES	0	-	**Reserved**
DCBYP	1	rw	**Data Cache/Data Read Buffer Bypass** 0_B DCache / DRB enabled 1_B DCache / DRB Bypassed (disabled)
RES	[31:2]	-	**Reserved**

Note: This register is endinit protected

Data Control Register 2 (DCON2)

DCON2 contains size information for the Data memory system.

DCON2
Data Control Register 2 (CSFR_Base + 9000_H) Reset Value: CPU Dependent



Field	Bits	Type	Description
DCACHE_SZE	[15:0]	r	**Data Cache Size** In KBytes
DSCRATCH_SZE	[31:16]	r	**Data Scratch Size** In KBytes

DMI Synchronous Trap Flag Register

The DSTR contains synchronous trap information for the data memory system. The register is updated with trap source information to aid the localisation of faults.

The register is updated whenever a valid trap is detected and the register has no bits already set. It is cleared by a write (independent of data value).

DSTR
Data Synchronous Trap Register

(CSFR_Base + 9010_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES							ALN	RES			CLE	MPE	CAC	SCE	SDE
r							rwh	r			rwh	rwh	rwh	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOE	DTME	RES							CRE	RES			LBE	GAE	SRE
rwh	rwh	r							rwh	r			rwh	rwh	rwh

Field	Bits	Type	Description
SRE	0	rwh	**Scratch Range Error** Data access to data scratch memory region outside of physically implemented memory
GAE	1	rwh	**Global Address Error** Load or store to local code scratch address outside of the lower 1MByte
LBE	2	rwh	**Load Bus Error** Data load from bus causing error
RES	[5:3]	r	**Reserved**
CRE	6	rwh	**Cache Refill Error** Bus error during cache refill
RES	[13:7]	r	**Reserved**
DTME	14	rwh	**DTAG MSIST Error** Access to memory mapped DTAG range outside of physically implemented memory
LOE	15	rwh	**Load Overlay Error** Load to invalid overlay address

Field	Bits	Type	Description
SDE	16	rwh	**Segment Difference Error** Load or store access where base address is in different segment to access address
SCE	17	rwh	**Segment Crossing Error** Load or store access across segment boundary
CAC	18	rwh	**CSFR Access Error** Load or store to local CSFR space
MPE	19	rwh	**Memory Protection Error** Data access violating memory protection.
CLE	20	rwh	**Context Location Error** Context operation to invalid location
RES	[23:21]	r	**Reserved**
ALN	24	rwh	**Alignment Error** Data access causing alignment error
RES	[31:25]	r	**Reserved**

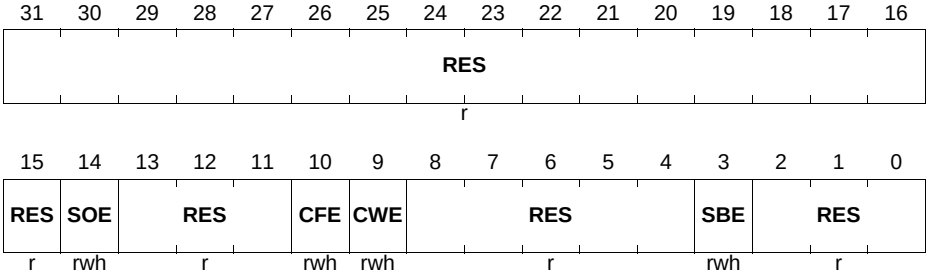
DMI Asynchronous Trap Flag Register

The DATR contains asynchronous trap information for the data memory system. The register is updated with trap information for DAE traps to aid the localisation of faults.

The register is updated whenever a valid trap is detected and the register has no bits already set. It is cleared by a write (independent of data value).

DAE traps are inhibited if the DATR register is non-zero.

DATR
Data Asynchronous Trap Register
(CSFR_Base + 9018_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RES	[2:0]	r	**Reserved**
SBE	3	rwh	**Store Bus Error** Data store to bus causing error
RES	[8:4]	r	**Reserved**
CWE	9	rwh	**Cache Writeback Error** Bus error during cache writeback operation
CFE	10	rwh	**Cache Flush Error** Bus error during cache flush operation
RES	[13:11]	r	**Reserved**
SOE	14	rwh	**Store Overlay Error** Store to invalid overlay address
RES	[31:15]	r	**Reserved**

Data Error Address Register

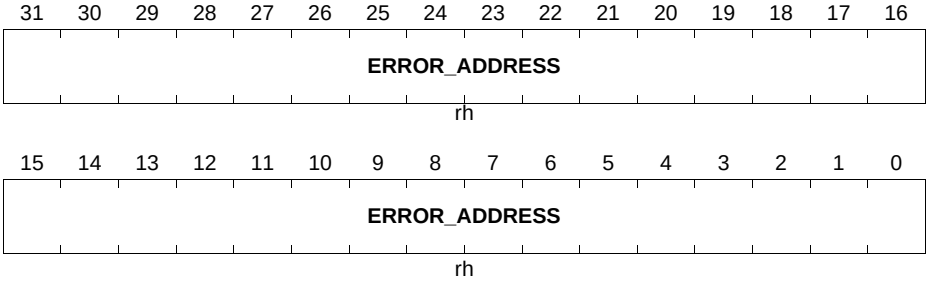
DEADD contains trap address information for the Data memory system. The register is updated with trap information for MEM, ALN, DSE or DAE traps to aid the localisation of faults.

The register is only set whenever a trap is detected and either the DATR or DSTR registers have no bits already set.

The register contents are only valid when either the DATR or DSTR register is non-zero and hence should be read prior to clearing these registers.

DEADD

Data Error Address Register (CSFR_Base + 901C$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
ERROR_ADDRESS	[31:0]	rh	**Error Address**

6.12 Safety Features

The CPUs implements a number of safety concepts These are detailed in the following sections.

6.12.1 SRI Data Master Address Phase Error Injection

To allow the SRI address phase error detection system to be tested it is necessary to inject errors during the address phase of an SRI transaction. This is done by the TC1.6P or TC1.6E data master. The data master selectively inverts individual bits of the address phase of an SRI read or write phase ECC packet and is controlled using the SEGEN register. The SEGEN register contains three fields:- an enable (AE), an error descriptor (ADTYPE) and a bit flip (ADFLIP) field. When enabled for an address phase error the address ECC bits indicated by the flip field are inverted for the next SRI data read or write bus transaction performed by the DMI. Following the transaction the enable bit is cleared by hardware. This mechanism allows selected bits of the SRI address ECC to be corrupted for a single transaction. The SEGEN register is ENDINIT protected.

6.12.2 SRI Data Master Write Phase Error Injection

To allow the SRI data phase error detection system to be tested it is necessary to inject errors during the data phase of an SRI write transaction. This is done by the TC1.6P or TC1.6E data master. The data master selectively inverting individual bits of the SRI write phase ECC packet and is controlled using the SEGEN register. The SEGEN register contains three fields:- an enable (AE), an error descriptor (ADTYPE) and a bit flip (ADFLIP) field. When enabled for a data phase error the data phase ECC bits indicated by the flip field are inverted for the next SRI write bus transaction performed by the DMI. Following the transaction the enable bit is cleared by hardware. This mechanism allows selected bits of the SRI data ECC to be corrupted for a single transaction. The SEGEN register is ENDINIT protected.

6.12.3 SRI Data Slave Read Phase Error Injection

To allow the SRI data phase error detection system to be tested it is necessary to inject errors during the data phase of an SRI read transaction. This is done by the TC1.6P or TC1.6E slave. The data slave selectively inverts individual bits of the SRI read phase ECC packet and is controlled using the SEGEN register. The SEGEN register contains three fields:- an enable (AE), an error descriptor (ADTYPE) and a bit flip (ADFLIP) field. When enabled for a data phase error the data phase ECC bits indicated by the flip field are inverted for the next SRI read bus transaction performed by the DMI. Following the transaction the enable bit is cleared by hardware. This mechanism allows selected bits of the SRI data ECC to be corrupted for a single transaction. The SEGEN register is ENDINIT protected.

6.12.4 SRI Error Capture

The SRI master and slave interfaces of the CPU implement the standard SRI ECC system. Error information detected during SRI transactions at the SRI master and slave interfaces is captured in the PIETR and PIEAR, or DIETR and DIEAR registers. On detection of an error the error information is captured and the relevant IED bit is set. No further error information is captured until this bit is cleared by software.

Error conditions at the SRI program interfaces are notified to the Safety Monitor Module via the pbus_err_o output from the CPU.

Error conditions at the SRI data interfaces are notified to the Safety Monitor Module via the dbus_err_o output from the CPU.

If an error is detected at an SRI slave interface during a write operation then the erroneous data will not be written.

6.12.5 SRI Safe Data Master tag

In order to differentiate between data accesses from safe and regular tasks a new safe task identification bit is introduced into the PSW register (PSW.S).

An SRI data access performed by a task when the PSW.S bit is set uses the safe data master tag. When this bit is not set the regular data master tag is used for the access. There is only one program master tag. This is used for SRI program fetches by both safe and regular tasks.

The initial value of the PSW.S bit for interrupt handlers is defined by the SYSCON.IS bit. The initial value of the PSW.S bit for trap handlers is defined by the SYSCON.IT bit.

On a trap the save of the current context is performed using the data master tag defined by SYSCON.TS

On an interrupt the save of the current context is performed using the data master tag defined by SYSCON.IS

6.12.6 Safety Memory Protection

The CPUs each allow for the definition of eight, protected regions of local SRAM memory.

The protection applies only to write or read-modify-write accesses to local SRAM included in the DMI or PMI from the bus.

Read accesses are not protected.

The protection scheme is based on the use of SRI tags to identify the master attempting the access and allows for a six bit tag individually identifying up to 64 masters. 32 masters are supported in the current implementation.

Each region is defined using the registers, SPROT_RGNLAx (x=0-7) to define the lower address of the region, SPROT_RGNUAx (x=0-7) to define the upper address of the

region and SPROT_RGNACCENx (x=0-7) to individually select the master tags permitted write access to the defined address range.

A write or read-modify-writes access is seen as valid if the master tag of the access is enabled in the SPROT_RGNACCENx register and the address of the access satisfies the following relationship:-

SPROT_RGNLAx <= address < SPROT_RGNUAx

The SPROT_RGNLAx, SPROT_RGNUAx, SPROT_RGNACCENx registers are protected by the safety_endinit signal.

After reset, the region address registers will be set to include the whole of the CPUs SRAM address space and write access by all masters will be enabled.

When altering protection settings, it should be noted that, due to access pipelining and resynchronization delays in the register block, a write to a memory address affected by the protection change occurring immediately after the register write initiating the change may, or may not, be affected by the changed settings.

Whenever a Safety Memory Protection violation occurs the event is notified to the Safety Management Unit via the tc16_safe_prot_err_o output signal from the CPU. The PIETR/DIETR and PIEAR/DIEAR registers are updated to aid localisation of the error

6.12.7 Safety Register Protection

The CPUs implement the standard memory protection scheme for peripheral registers using the SPROT_ACCEN register. This allows all CPU CSFR and SFR registers to be protected from corruption by untrusted masters. The SPROT_ACCEN register defines which masters may modify the SFR and CSFR registers via bus access through the SRI slave interface.

The SPROT_ACCEN register is protected by the safety_endinit signal.

In all cases the master is identified using the SRI tag of the access. See On Chip Bus chapter for the product's TAG ID to master peripheral mapping.

Whenever a Safety Register Protection violation occurs the event is notified to the Safety Management Unit via the safe_prot_err_o output signal from the CPU. The PIETR and PIEAR registers are updated to aid localisation of the error.

6.12.8 Lock Step Implementation

The TC1.6E and TC1.6P CPUs may be implemented as standard CPU (as described in the chapter) or as a checker CPU for use in lockstep comparison system. The checker CPU implementation has the following features.

- No Memories

- All signals that would normally be fed to the memories become additional outputs from the checker CPU.
- All signals that would normally be fed from the memories become additional inputs to the checker CPU.
- All outputs from the checker CPU are inverted.

The lockstep system compares the following CPU interfaces.

- Interface to the interrupt router.
- Interface to the SCU.
- Interface to the DMI Data Scratch pad memory.
- Interface to the PMI Program Scratch memory.
- Interface to the PMI Program Cache memory.
- Interface to the PMI Program Tag memory.
- Master interface to the Peripheral bus.
- Master interface to the SRI bus for PMI accesses.
- Master interface to the SRI bus for DMI accesses.
- Slave accesses from the SRI bus for PMI and DMI accesses.
- The CPUs run in lockstep mode at the full specified frequency. Enabling lockstep does not require operational frequency to be reduced.
- The Checker CPU is implemented using inverse state storage “AntiCore”.

6.12.9 MBIST usage recommendations

The following usage of the CPU SRAM MBIST system is recommended.

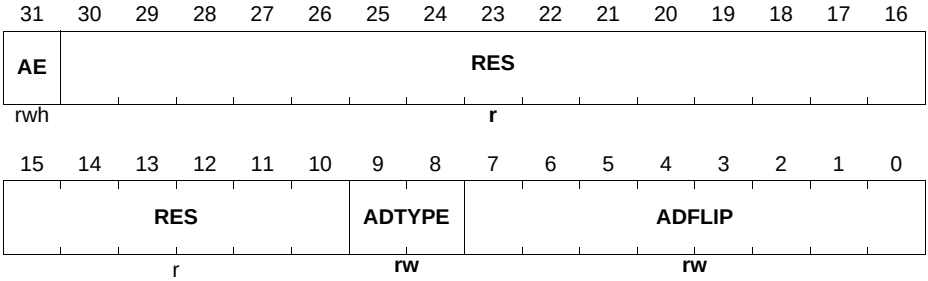
- When any one of the CPU local data memories (DSPR/DCACHE or DTAG) is in MBIST test mode (MEMTEST enabled), then other CPU local data memories will be not be accessible. It is therefore recommended to enable DSPR/DCACHE MBIST mode and DTAG memory MBIST mode together.
- When any one of the CPU local program memories (PSPR/PCACHE or PTAG) is in MBIST test mode (MEMTEST enabled), then other CPU local program memories will be not be accessible. It is therefore recommended to enable PSPR/PCACHE MBIST mode and PTAG memory MBIST mode together.
- For configurations that use dual MBIST controllers for a single logical memory (e.g TC29x DSPR/DCACHE for CPU1,2), MBIST tests run on one CPU will report errors to both controllers. It is therefore recommended to enable MBIST mode on both controllers together and that the error notification bits of both controllers are cleared at the end of testing.

6.12.10 Registers Implementing Safety Features

SRI Error Generation Register

The SEGEN register controls the injection of SRI address phase errors from the DMI. This register is ENDINIT protected.

SEGEN
SRI Error Generation Register (CSFR_Base + 1030_H) **Reset Value: $0000\ 0000_H$**

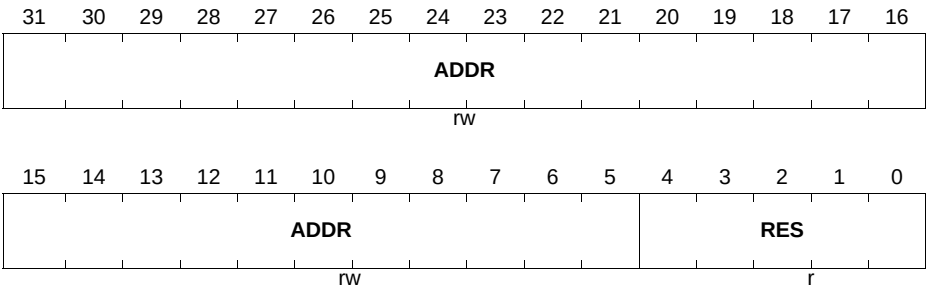


Field	Bits	Type	Description
ADFLIP	[7:0]	rw	**Address ECC Bit Flip** SRI address ECC Bits to be flipped on the next read or write transaction from the DMI when enabled by AE. 0_B No Flip 1_B Flip
ADTYPE	[9:8]	rw	**Type of Error** 00_B Data Master Address Phase 01_B Data Master Write Data 10_B Data Slave Read Data 11_B Reserved
RES	[30:10]	r	**Reserved**
AE	31	rwh	**Activate Error Enable** Enabled the selective inverting of SRI ECC packet bits defined by ADFLIP. This bit will be cleared by hardware after the next SRI read or write transaction from the DMI. 0_B Not Enabled 1_B Enabled

Safety Protection Region Lower Address Register

In conjunction with the associated SPROT_RGNUAx and SPROT_RGNACCENx registers, the SPROT_RGNLA register provides control of a memory protection region. SPROT_RGNLAx defines the lower address of a region of memory, SPROT_RGNUAx defines the upper address and the SPROT_RGNACCENx registers define the SRI tags allowed write access to the region. Address ranges can bet set to be larger than the SRAM address space but only accesses to the SRAM are affected by these registers. The minimum resolution of the comparison logic is 32_D bytes so address bits 4_D down to 0_D are not used

SPROT_RGNLAx (x=0-7)
Safety Protection Region Lower Address Register
(SFR_Base + $E000_H$+x*10_H) Reset Value: 0000 0000_H

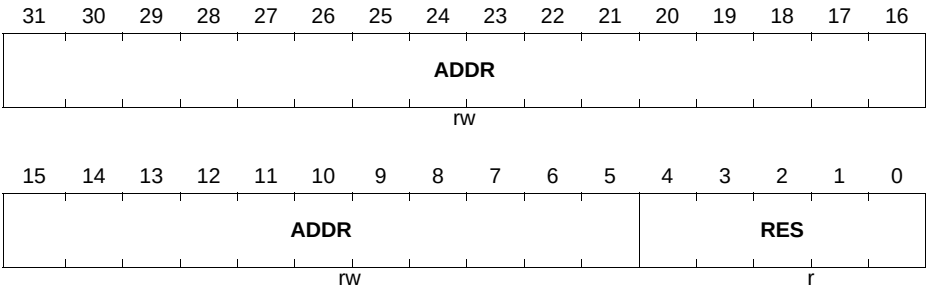


Field	Bits	Type	Description
RES	[4:0]	r	**Reserved**
ADDR	[31:5]	rw	**Region Lower Address** Bits 32 to 5 of the address which is the lower bound of the defined memory region

Safety protection Region Upper Address Register

In conjunction with the associated SPROT_RGNLAx and SPROT_RGNACCENx registers, the SPROT_RGNUAx register provides control of a memory protection region. SPROT_RGNUA defines the upper address of the memory region

SPROT_RGNUAx (x=0-7)
Safety protection Region Upper Address Register
(SFR_Base + $E004_H$+x*10_H) Reset Value: FFFF $FFE0_H$



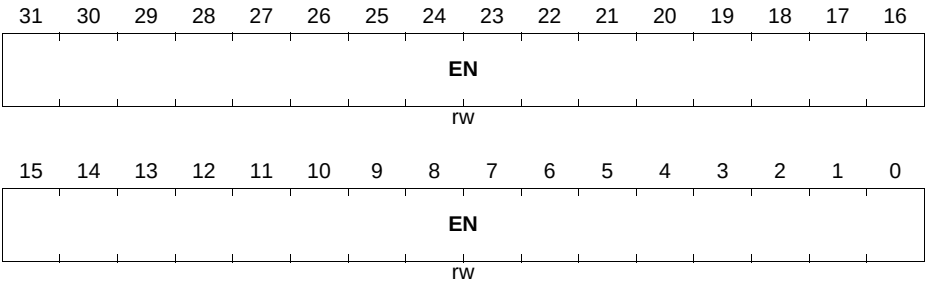
Field	Bits	Type	Description
RES	4:0	r	**Reserved** Unused bits will always read as 0_B. Written value will be ignored
ADDR	31:5	rw	**Region Upper Address** Bits 31 to 5 of the address which is the upper bound of the defined memory region

Safety Region Access Enable Register A

The Access Enable Register A controls write access for transactions to the protected memory region with the on chip bus master TAG ID 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The Safety protection is prepared for an 6 bit TAG ID. The registers SPROT_RACCEN0 / SPROT_RACCEN1 are provide one enable bit for each possible 6 bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000_B, EN1 -> TAG ID 000001_B, ... ,EN31 -> TAG ID 011111_B.

SPROT_RGNACCENAx (x=0-7)
Safety Protection Region Access Enable Register A
(SFR_Base+ $E008_H$+x*10_H) Reset Value: FFFF $FFFF_H$



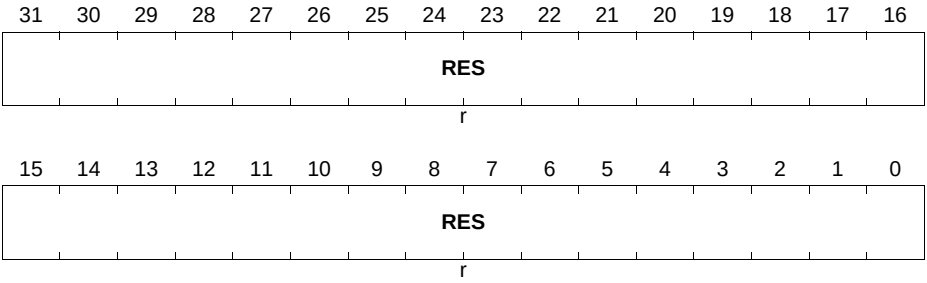
Field	Bits	Type	Description
EN	[31:0]	rw	**Access Enable for Master TAG ID n (n=0-31)** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Safety Protection Region Access Enable Register B

The Access Enable Register B controls write access for transactions to the protected memory region with the on chip bus master TAG ID 100000_B to 111111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The CPU is prepared for an 6 bit TAG ID. The registers SPROT_RACCEN0 / SPROT_RACCEN1 are providing one enable bit for each possible 6 bit TAG ID encoding.

Mapping of TAG IDs to SPROT_ACCENB.ENx: EN0 -> TAG ID 100000_B, EN1 -> TAG ID 100001_B, ... ,EN31 -> TAG ID 111111_B.

SPROT_RGNACCENBx (x=0-7)
Safety Protection Region Access Enable Register B (SFR_Base + $E00C_H$+x*10_H)
Reset Value: 0000 0000_H



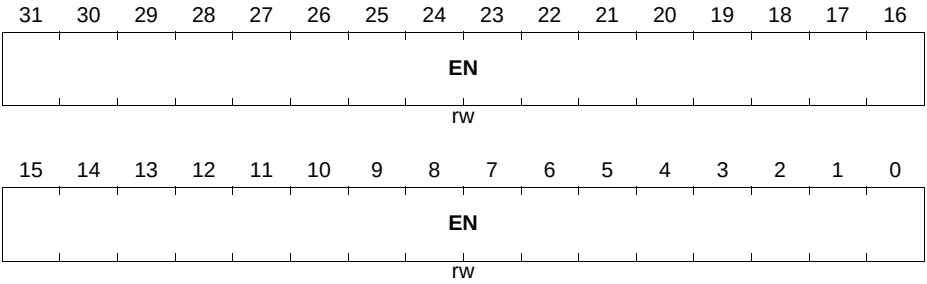
Field	Bits	Type	Description
RES	[31:0]	r	**Reserved** Read as 0; should be written with 0.

Safety Register Access Enable Register A

The Access Enable Register A controls write access for transactions to CSFR/SFR registers with the on chip bus master TAG ID 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The Safety protection is prepared for an 6 bit TAG ID. The registers SPROT_RACCEN0 / SPROT_RACCEN1 are provide one enable bit for each possible 6 bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000_B, EN1 -> TAG ID 000001_B, ... ,EN31 -> TAG ID 011111_B.

SPROT_ACCENA
Safety Protection Register Access Enable Register A
(SFR_Base + $E100_H$) **Reset Value: FFFF FFFF$_H$**



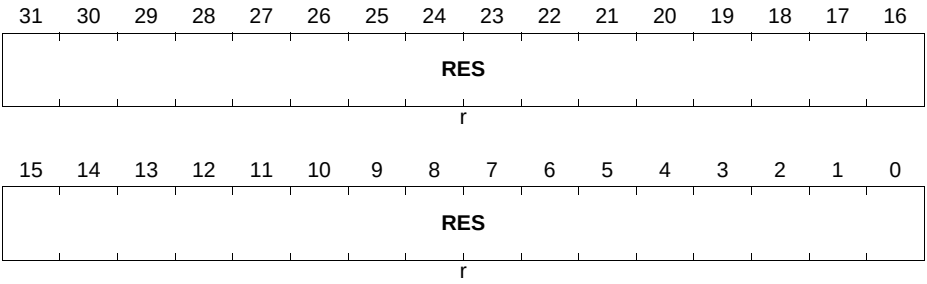
Field	Bits	Type	Description
EN	[31:0]	rw	**Access Enable for Master TAG ID n (n= 0-31)** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Safety Protection Register Access Enable Register B

The Access Enable Register B controls write access for transactions to CSFR/SFR registers with the on chip bus master TAG ID 100000_B to 111111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The CPU is prepared for an 6 bit TAG ID. The registers SPROT_RACCEN0 / SPROT_RACCEN1 are providing one enable bit for each possible 6 bit TAG ID encoding.

Mapping of TAG IDs to SPROT_ACCENB.ENx: EN0 -> TAG ID 100000_B, EN1 -> TAG ID 100001_B, ... ,EN31 -> TAG ID 111111_B.

SPROT_ACCENB
Safety Protection Region Access Enable Register B
(SFR_Base + E104$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RES	[31:0]	r	**Reserved** Read as 0; should be written with 0.

7 Lockstep Comparator Logic (LCL)

The Lockstep Comparator Logic module provides access to the control and self test functions of the processor lockstep comparators.

7.1 Feature List

An overview of the features implemented in the LCL follows:

- Monitoring the core comparators for the lockstep core and its shadow and flagging any detected differences
- Running background, continuous self test on the lockstep comparators to validate the correct operation of the logic.

7.2 Lockstep Control

The lockstep control function is enabled by the LSEN bitfield in a control register in the SCU. Each core capable of lockstep has its own instance of the control register. In this product, the CPU0 instance of the Tricore can be lockstepped so there is one register, LCLCON0 for CPU0.

These registers are only initialised by a cold power-on reset. In this initialisation state, all lockstepped processors in the system will have lockstep enabled. The lockstep function can only be disabled by the system initialisation software writing a 0_B to the LSEN bitfield. Application software cannot enable or disable the lockstep function.

The current mode of the lockstep logic can be monitored by reading the lockstep status bit, LS, in the associated LCLCON register.

Writes to the control registers will be subject to the protection mechanisms of the SCU.

7.3 Lockstep Monitoring

The lockstep monitoring function will compare the outputs from the master and checker cores and report that a failure has occurred to the Safety Management Unit (SMU) for appropriate action.

The monitoring function temporally separates the cores by inserting synchronisation delays to re-align the signals being compared. To achieve this, the checker core inputs and the master core outputs fed to the comparators are delayed by two clock cycles.

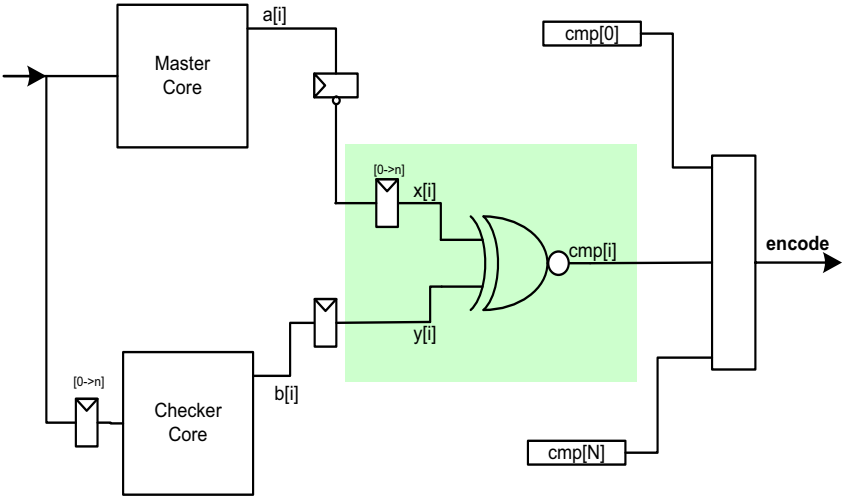


Figure 7-1 Node Comparator

Figure 7-1 above shows the equivalent circuit of an arbitrary node comparator, i, of 0 to N comparators. The comparator monitors two signals, **a** and **b**, which are connected to the same node in the master and checker cores. The signals can be synchronised in the relevant core using the core clock to minimise impact on the core timing.

After the optional synchronisation, the master core monitor point is inverted to reduce the risk of a common mode failure in the two monitored signals. Therefore, **x** is equivalent to **a** delayed by a n clock cycles, where n=2 in this implementation, to allow for the temporal shift between the master and checker cores and **y** is equivalent to **b**. If the nodes differ (i.e. **x** and **y** are the same), the CMP signal is set to 1_B. This will cause the **encode** signal to flag the failing node and the failure will be detected. In the event that multiple nodes fail, the encode output will be the minimum and maximum values of all the indices, **i**, of the failing nodes. This has no impact on normal functioning but allows the self test logic to detect a real failure occurring in the same clock cycle that a fault is injected for test purposes.

7.4 Lockstep Self Test

Each core capable of lockstep also has a continuously running background self test of the lockstep comparator.

The self test function will inject faults into both inputs of each of the monitored nodes and verify that the fault is correctly detected by the monitoring logic.

In the event of a self test failure being detected, the failure will be reported to the Safety Management Unit for an appropriate response.

An equivalent circuit of a node comparator showing the fault injection logic is shown in **Figure 7-2**

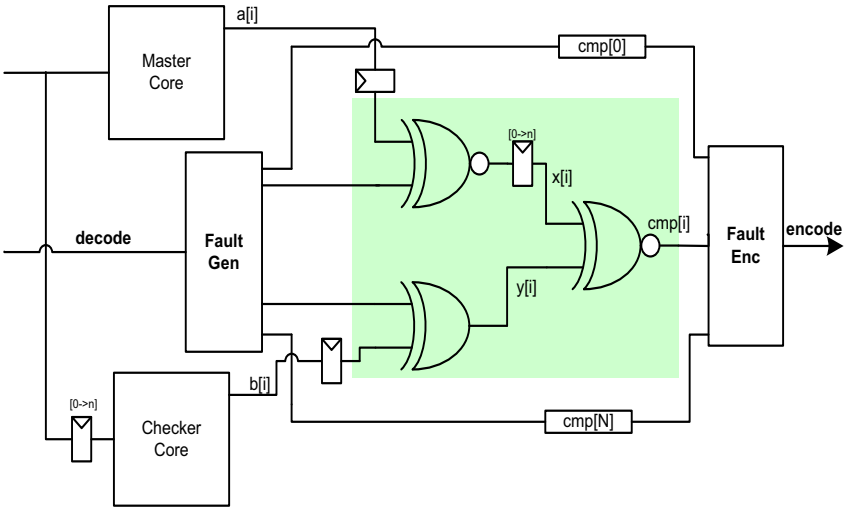


Figure 7-2 Node Comparator with Fault Injection

Faults are injected using the **Fault Gen** block. This will use a binary number, **decode**, to calculate which node is to be tested. **decode** will be generated from a free running, binary counter using gray code number representation (the **input counter**). The Self test circuit will test every node once every 8192 clock cycles. Alternate test cycles will inject faults into either the **a** or **b** side of the comparator node. The complete self test cycle will therefore repeat every 16384 clock cycles.

Injecting a fault into either side of the comparator node will cause that node to fail unless a real fault occurs in the same clock cycle in which case the two faults will cancel out.

The node failing is processed by the **Fault Enc** block to generate a binary representation of the failing node, **encode**. This contains two numbers, the node index of the first node found to be failing counting up from the lowest minimum node index and the node index of the first node found to be failing counting down from the maximum node index.

If the lockstep block is functioning correctly, both values in **encode**, will either be 0 if no failures have been detected or the number of the node which has had a fault induced by the self test logic.

The values in **encode** are checked against a second, independent binary counter (the **monitor counter**). The **monitor counter** is also compared against the value of the **input counter**. In the event that either of the values in **encode** or the **input counter** fails to match the value of the monitor counter, a failure condition will be flagged to the SMU.

With this implementation, any of the following conditions will cause a self test fail:

- an actual fault occurring on any of the **a** or **b** nodes
- a stuck at 0_B fault occurring on any of the **cmp** nodes
- a stuck at 1_B fault occurring on any of the **cmp** nodes
- a failure in the **Fault Gen** block causing an incorrect or no fault to be injected
- a failure in the **Fault Enc** block causing an incorrect detection or no fault to be detected
- a stuck at fault on **x** (assuming that an injected fault does not always coincide with a masking pulse on **b**)
- a stuck at fault on **y** (assuming that an injected does not always coincide with a masking pulse on **a**)
- an actual fault on any of the **a** or **b** nodes coinciding with an injected fault
- a soft error occurring on either the **input counter** or **monitor counter**.

7.5 Lockstep Failure Signalling Test

The lockstep comparator allows a failure to be injected into one of the comparator nodes to allow the signalling of failures to be verified. A failure can be injected by writing 1_B to the LCLT0 bitfield of the LCLTEST register. The failure will be injected for a single cycle of the SPB clock. It is not necessary to write a 0_B to clear the test.

7.6 Functional Redundancy

All registers in the lockstep block which are not capable of being directly monitored for correct operation by the self test function will be duplicated. In the event of the duplicated registers not storing the same state, an error will be flagged to the SMU.

7.7 Revision History

Deltas from Lockstep version 1.1 to version 1.2

- Bitfield names updated for LCLCON register. See **sections “Lockstep Monitoring” and “Lockstep Self Test”.**
- Multiple failure detecton algorithm updates. See Lockstep Self Test section.

Deltas from Lockstep version 1.2 to 1.3

- Revision History Added

Deltas from Lockstep version 1.3 to 1.4

- Description of self test updated

Deltas from Lockstep version 1.4 to 1.5

- Control bits for self test and delay removed. Both functions will now be permanently enabled.
- Lockstep will now be on after reset and cannot be enabled once switched off

Deltas from Lockstep version 1.5 to 1.6

- Additional control bit added to allow injection of fault condition under software control.

Deltas from Lockstep version 1.6 to 1.7

- Specifying cold power-on reset / typo’s.

Deltas from Lockstep version 1.7 to 1.8

- No functional change, updates to Requirements tags only.

Deltas from Lockstep version 1.8 to 1.9

- No functional change, correction of typos and support for TC23x configuration.

8 System Control Units

The System Control Cluster comprises various modules which each handle system control functions.

The System Control Cluster includes the following functional modules:

- Clock Control Unit (CCU)
- Reset Control Unit (RCU)
- Power Management Controller (PMC)
- System Control Unit (SCU), including ERU,LCL, WDT, EMS, NMI and OVC registers

8.1 Clocking and Clock Control Unit (CCU)

This section describes the TC21x/TC22x/TC23x clock system, its configuration and the principles upon which it is based.

The clock system itself is build up as chain composing out of different building blocks which allow different certain function parts for this complete chain.

Building blocks are:

- Basic clock generation (Clock Source)
- Clock speed upscaling
- Clock distribution
- Individual clock configuration

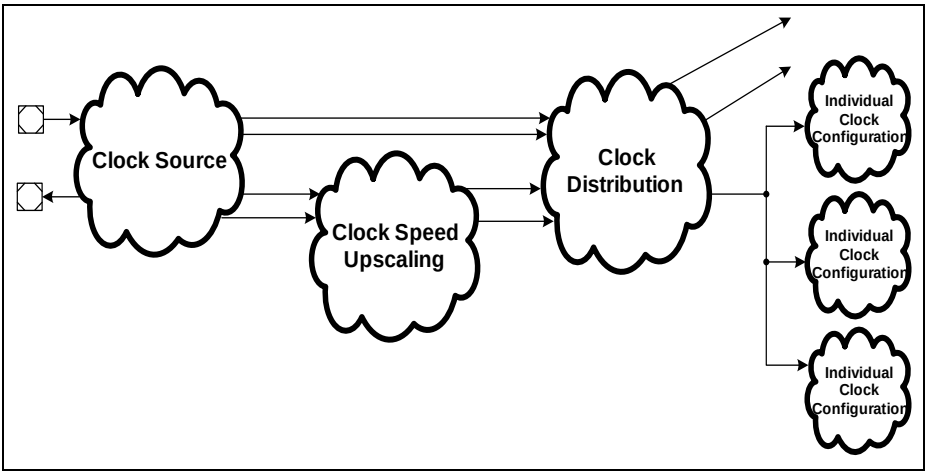


Figure 8-1 Clock Tree

Aside from the pure clock generation options there are several support functions which have been integrated in order to support easy and convenient controls.

In the following subsections the clock tree is explained from left to the right. Using this flow for the initial configuration is also recommended. Expert users can of course execute the configuration in a individual order.

Note: If a running system needs to be reconfigured not all parts of the clock tree need to be configured, only these parts that are required can be updated.

8.1.1 Clock Sources

There are several clock sources available which either source the complete device, a major or minor part, or only dedicated modules. This depends on the sources and the configuration.

Please note that several clock sources can be used in parallel inside the system but the main function of each peripheral is only related to one source at any time.

8.1.1.1 Oscillator Circuit (OSC)

The oscillator circuit, a Pierce oscillator, is designed to work with both an external crystal / ceramic resonator or an external stable clock source, consists of an inverting amplifier with XTAL1 as input, and XTAL2 as output with an integrated feedback resistor.

External Input Clock Mode

When using an external clock signal it must be connected to XTAL1. XTAL2 is left open (unconnected).

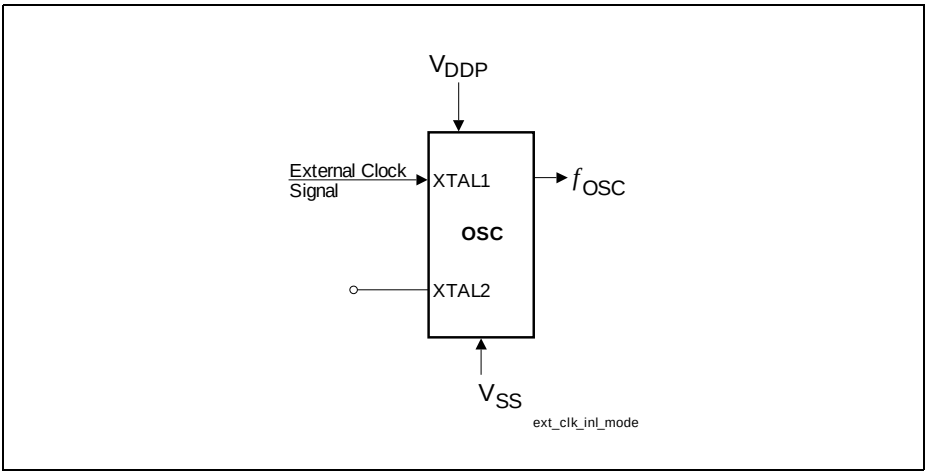


Figure 8-2 TC21x/TC22x/TC23x Direct Clock Input

When supplying the clock signal directly, not using an external crystal / ceramic resonator and bypassing the oscillator, the input frequency needs to be equal or greater than PLL VCO input frequency (the value is listed in the Data Sheet) if used in normal Mode.

External Crystal / Ceramic Resonator Mode

Figure 8-3 shows the recommended external circuitries for both operating modes, External Crystal / Ceramic Resonator Mode with and without external components.

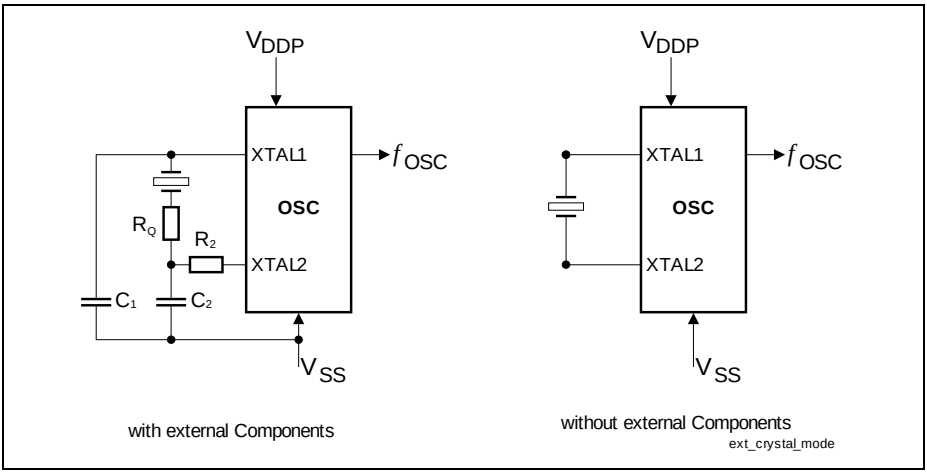


Figure 8-3 External Circuitry for Crystal / Ceramic Resonator operation

When using an external crystal / ceramic resonator, its frequency can be within the allowed range (the values are listed in the Data Sheet). An external oscillator load circuitry can be used, connected to both pins, XTAL1 and XTAL2. Additionally are necessary, two load capacitances C_1 and C_2 (see **Figure 8-3**), and depending on the crystal / ceramic resonator type, a series resistor R_2 to limit the current. A test resistor R_Q may be temporarily inserted to measure the oscillation allowance (negative resistance) of the oscillator circuitry. R_Q values are typically specified by the crystal / ceramic resonator vendor. The C_1 and C_2 values shown in the Data Sheet can be used as starting points for the negative resistance evaluation and for non-productive systems. The exact values and related operating range are dependent on the crystal / ceramic resonator frequency and have to be determined and optimized together with the crystal / ceramic resonator vendor using the negative resistance method. Oscillation measurement with the final target system is strongly recommended to verify the input amplitude at XTAL1 and to determine the actual oscillation allowance (margin negative resistance) for the oscillator-crystal / ceramic resonator system.

The oscillator can also be used in combination with a ceramic resonator. The final circuitry must be also verified by the resonator vendor.

Oscillator Circuit Control Register

OSCCON
OSC Control Register **(010_H)** **Reset Value: 0000 0X1X$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				0				APR EN	0		OSCVAL				
r				rw				rw	r		rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				X1D EN	X1D	0	PLL HV	SH BY	MODE		GAINSEL		OSC RES	PLL LV	0
r				rw	rh	r	rh	rw	rw		rw		w	rh	

Field	Bits	Type	Description
PLLLV	1	rh	**Oscillator for PLL Valid Low Status Bit** This bit indicates if the frequency output of OSC is usable for the VCO part of the PLL. This is checked by the Oscillator Watchdog of the PLL. 0_B The OSC frequency is not usable. Frequency f_{REF} is too low. 1_B The OSC frequency is usable *Note: The default value depends on the status of the clock received from pin XTAL1.*
OSCRES	2	w	**Oscillator Watchdog Reset** 0_B The Oscillator Watchdog of the PLL is not cleared and remains active 1_B The Oscillator Watchdog of the PLL is cleared and restarted *Note: Always read as zero.*

Field	Bits	Type	Description
GAINSEL	[4:3]	rw	**Oscillator Gain Selection** In Normal Mode this value should not be changed from the reset value 11_B. 00_B The gain control is configured for frequencies from 4 MHz to 8 MHz 01_B The gain control is configured for frequencies from 4 MHz to 16 MHz 10_B The gain control is configured for frequencies from 4 MHz to 20 MHz 11_B The gain control is configured for frequencies from 4 MHz to 25 MHz External Crystal / 40 MHz Ceramic Resonator *Note: Default value after reset is 11_B.*
MODE	[6:5]	rw	**Oscillator Mode** This bit field defines which mode can be used and if the oscillator entered the Power-Saving Mode or not. 00_B External Crystal / Ceramic Resonator Mode and External Input Clock Mode. The oscillator Power-Saving Mode is not entered. 01_B OSC is disabled. The oscillator Power-Saving Mode is not entered. 10_B External Input Clock Mode and the oscillator Power-Saving Mode is entered 11_B OSC is disabled. The oscillator Power-Saving Mode is entered. *Note: Default value after reset is 00_B.*
SHBY	7	rw	**Shaper Bypass** 0_B The shaper is not bypassed 1_B The shaper is bypassed *Note: Default value after reset is 0_B.*

Field	Bits	Type	Description
PLLHV	8	rh	**Oscillator for PLL Valid High Status Bit** This bit indicates if the frequency output of OSC is usable for the VCO part of the PLL. This is checked by the Oscillator Watchdog of the PLL. 0_B The OSC frequency is not usable. Frequency f_{OSC} is too high. 1_B The OSC frequency is usable *Note: The default value depends on the status of the clock received from pin XTAL1.*
X1D	10	rh	**XTAL1 Data Value** This bit monitors the value (level) of pin XTAL1. If XTAL1 is not used as clock input it can be used as GPI pin. This bit is only updated if X1DEN is set. *Note: The default value depends on the status of the clock received from pin XTAL1.*
X1DEN	11	rw	**XTAL1 Data Enable** 0_B Bit X1D is not updated 1_B Bit X1D can be updated *Note: Default value after reset is 0_B.*
OSCVAL	[20:16]	rw	**OSC Frequency Value** This bit field defines the divider value that generates the reference clock that is supervised by the oscillator watchdog. f_{OSC} is divided by OSCVAL + 1 in order to generate f_{OSCREF}.
APREN	23	rw	**Amplitude Regulation Enable** 0_B Amplitude Regulation is disabled 1_B Amplitude Regulation is enabled
0	[27:24]	rw	**Reserved** Should be written with 0.
0	0, 9, [15:12], [22:21], [31:28]	r	**Reserved** Read as 0; should be written with 0.

Configuration of the Oscillator

A configuration of the oscillator is always required before an external crystal / ceramic resonator could be used as clock source. For this start-up configuration two options are supported:

- Configuration via the SSW
- Configuration after the execution of the SSW

Configuration via SSW

This option is enabled when bit FLASH0_PROCOND.OSCCFG is set. In this mode the control information for the register OSCCON is loaded from FLASH0_PROCOND by the SSW. Therefore the required information needs to be stored with the UCB_DFlash at offset 00_H.

In this mode OSCCON.MODE and OSCCON.CAPxEN together with bit OSCCON.APREN are controllable.

Note: A control for OSCCON.GAINSEL is not required as it is recommended always to keep the default configuration of 11_B.

If the oscillator was disabled it must be enabled by setting bit field OSCCON.MODE = 00_B.

If the integrated capacitors should be used this can be enabled via the bits OSCCON.CAPxEN. If OSCCON[27:24] ≠ 0000_B then bit OSCCON.APREN need to be set too. Please note that the Amplitude Regulation needs to be enabled always when integrated caps are used. Enabling the Amplitude Regulation does not require a change for OSCCON.GAINSEL, which should be left at 11_B.

Configuration after SSW

After the optional configuration in the SSW there is always an option to configure the oscillator in the application software.

This is done via register OSCCON. The register itself is Safety ENDINIT protected.

In general the same rules apply for configuration as described in the section before.

Miscellaneous Oscillator Features

Support for two additional features which are related to the oscillator are also located in the OSCCON register.

XTAL1 as General Purpose Input Pin

For the rare case that XTAL1 / 2 are not used for clocking purpose XTAL1 could be used as an input pad instead. Setting bit OSCCON.X1DEN enables this function and the input value can be read thereafter via bit OSCCON.XD1.

Please note that this defines only a basic input function and supports no output features for XTAL1 or XTAL2.

Note: For XTAL2 no input function is supported.

Note: If XTAL1 is used for clock functions bit OSCCON.X1DEN should be cleared, as otherwise bit OSCCON.X1D would toggle with the input clock frequency.

Oscillator Watchdog

In combination with the PLL, a monitoring function is implemented. This feature is defined to detect severe malfunctions of an external crystal / ceramic resonator. Detectable errors are loss of clock input or a much too high input frequency.

The oscillator watchdog monitors the incoming clock frequency f_{OSC} from OSC. A stable and defined input frequency is a mandatory requirement for operation in both Prescaler Mode and Normal Mode of the PLLs. For operation in Freerunning Mode no f_{OSC} input frequency is required. Therefore this mode is selected automatically after each System Reset. In addition for the Normal Mode it is required that the input frequency f_{OSC} is in a certain frequency range to obtain a stable master clock from the VCO part.

The expected input frequency is selected via the bit field OSCCON.OSCVAL. The OSC_WDT checks for frequencies which are too low or too high.

The frequency that is monitored is f_{OSCREF} which is derived from f_{OSC}.

(8.1)

$$f_{OSCREF} = \frac{f_{OSC}}{OSCVAL + 1}$$

The divider value OSCCON.OSCVAL has to be selected in a way that f_{OSCREF} is 2.5 MHz.

Note: f_{OSCREF} has to be within the range of 2 MHz to 3 MHz and should be as close as possible to 2.5 MHz.

Before configuring the OSC_WDT function all the SMU Oscillator Watchdog alarm response options should be disabled in order to avoid unintended traps. Thereafter the value of OSCCON.OSCVAL can be changed. Then the OSC_WDT should be reset by setting OSCCON.OSCRES. This requests the start of OSC_WDT monitoring with the new configuration. When the expected positive monitoring results of OSCCON.PLLLV and / or OSCCON.PLLHV are set the input frequency is within the expected range. As setting OSCCON.OSCRES clears both bits OSCCON.PLLLV and OSCCON.PLLHV all both status flags will be set. Therefore both flags should be cleared before the SMU

alarm responses are enabled again. The trap disabling-clearing-enabling sequence should also be used if only bit OSCCON.OSCRES is set without any modification of OSCCON.OSCVAL.

The oscillator clock is selected as source for the watchdog by configuring CCUCON1.INSEL = 01_B.

8.1.1.2 Back-up Clock

A back-up clock source is available as alternate clock source. This clock source provides a stable but reliable clock source that can be used as clock for the system. It provides less accuracy than an external crystal or ceramic resonator. The back-up clock can not be enabled or disabled or anyhow else be controlled. Therefore no control bits beside the selection itself as source (CCUCON0.CLKSEL = 00_B as clock source for the clock distribution and CCUCON1.INSEL = 00_B as clock source for the PLL / PLL_ERAY) are available.

8.1.2 Clock Speed Upscaling

Typical CPU operating speeds are about 10 times higher than the speed of the used crystal as clock source. Therefore an upscaling of the clock frequency is required.

For the upscaling up to two Phase Lock Loop (PLLs) are provided.

8.1.2.1 Phase-Locked Loop (PLL) Module

The PLL can convert a low-frequency external clock signal to a high-speed internal clock for maximum performance. It allows the use of input and output frequencies of a wide range by varying the different divider factors.

The PLL also has fail-safe logic that detects degenerate external clock behavior such as abnormal frequency deviations or a total loss of the external clock. It can execute emergency actions if it loses its lock on the external clock.

Features

- VCO lock detection
- 4-bit input divider **P:** (divide by PDIV+1)
- 7-bit feedback divider **N**: (multiply by NDIV+1)
- 7-bit output divider **K1 or K2:** (divide by either by K1DIV+1 or K2DIV+1)
- 7-bit output divider **K3:** (divide by K3DIV+1)
- Oscillator Watchdog
  - Detecting too low input frequencies
  - Detecting too high input frequencies
- Different operating modes

  - Prescaler Mode
  - Freerunning Mode
  - Normal Mode
- VCO Power Down
- Glitchless switching between both K-Dividers
- Glitchless switching between Normal Mode and Prescaler Mode
- Frequency Modulation
- Jitter reduction for modulation jitter

PLL Functional Description

The following figure shows the PLL block structure.

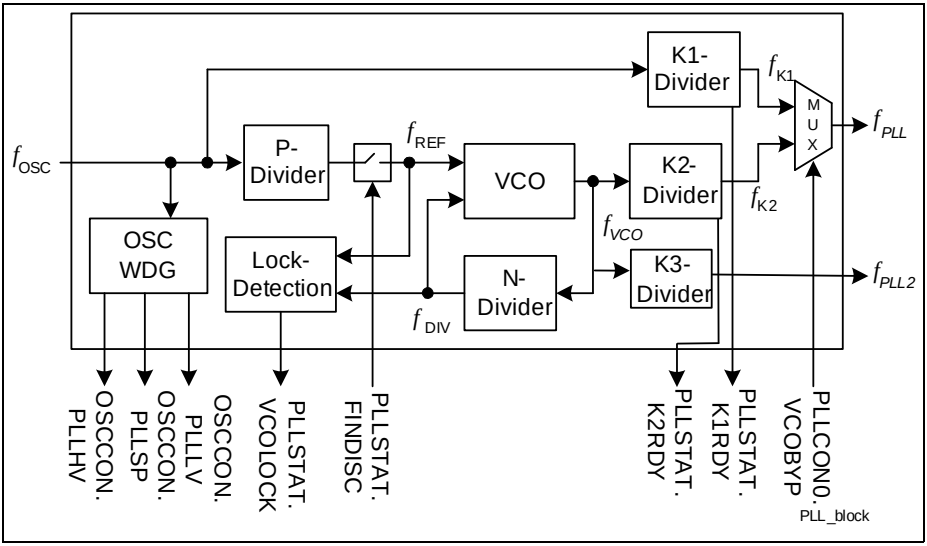


Figure 8-4 PLL Block Diagram

Mode Control

The PLL clock f_{PLL} and f_{PLL2} are generated from f_{OSC} in one of following software selectable modes:

- Normal Mode
- Prescaler Mode
- Freerunning Mode

Normal Mode

In Normal Mode the input frequency f_{OSC} is divided down by a factor P, multiplied by a factor N and then divided down by a factor K2 or K3.

The output frequency is given by

$$f_{PLL} = \frac{N}{P \cdot K2} \cdot f_{OSC} \quad (8.2)$$

$$f_{PLL2} = \frac{N}{P \cdot K3} \cdot f_{OSC} \quad (8.3)$$

Prescaler Mode

In Prescaler Mode the reference frequency f_{OSC} is only divided down by a factor K1.

The output frequency is given by

$$f_{PLL} = \frac{f_{OSC}}{K1} \quad (8.4)$$

Freerunning Mode

In Freerunning Mode the base frequency output of the Voltage Controlled Oscillator (VCO) $f_{PLLBASE}$ is only divided down by a factor K2 or K3.

The output frequency is given by

$$f_{PLL} = \frac{f_{PLLBASE}}{K2} \quad (8.5)$$

$$f_{PLL2} = \frac{f_{PLLBASE}}{K3} \quad (8.6)$$

Configuration and Operation of the Freerunning Mode

In Freerunning Mode, the PLL is running at its VCO base frequency and f_{PLL} / f_{PLL2} is derived from f_{VCO} only by the K2 / K3-Divider.

The Freerunning Mode is entered after each System Reset.

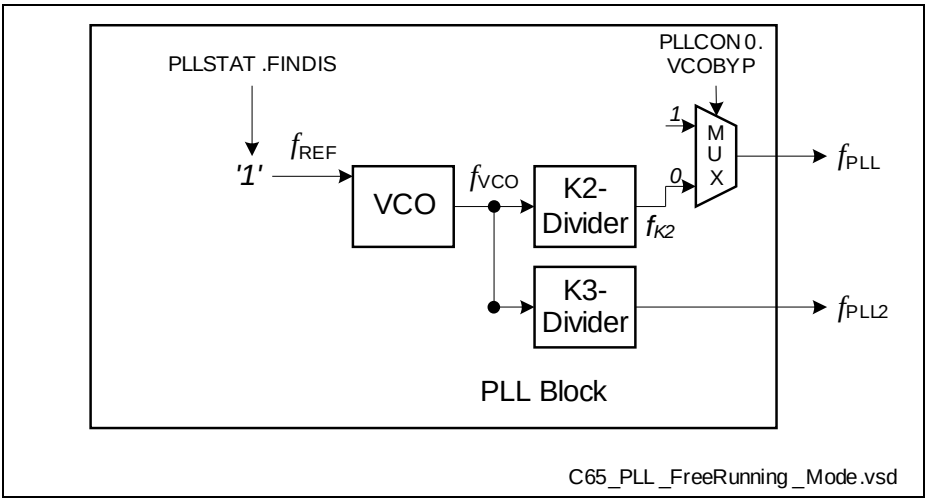


Figure 8-5 PLL Free-Running Mode Diagram

The output frequency is given by

$$f_{PLL} = \frac{f_{PLLBASE}}{K2} \quad (8.7)$$

$$f_{PLL2} = \frac{f_{PLLBASE}}{K3} \quad (8.8)$$

The Freerunning Mode is selected by the following settings

- PLLCON0.VCOBYP = 0
- PLLCON0.SETFINDIS = 1

The Freerunning Mode is entered when

- PLLSTAT.FINDIS = 1
- AND
- PLLSTAT.VCOBYST = 0

Operation on the Freerunning Mode does not require an input clock frequency of f_{OSC}. The Freerunning Mode is automatically entered on a PLL VCO Loss-of-Lock event if bit PLLCON0.OSCDISCDIS is cleared. This mechanism allows a fail-safe operation of the PLL as in emergency cases still a clock is available.

The frequency of the Freerunning Mode $f_{PLLBASE}$ is listed in the Data Sheet.

Note: Changing the system operation frequency by changing the value of the K2-Divider has a direct effect on the power consumption of the device. Therefore this has to be changed carefully to avoid excessive current steps.

Depending on the selected divider value of the K2-Divider the duty cycle of the clock is selected. This can have an impact for the operation with an external communication interface.

Configuration and Operation of the Prescaler Mode

In Prescaler Mode, the PLL is running at the external frequency f_{OSC} and f_{PLL} is derived from f_{OSC} only by the K1-Divider.

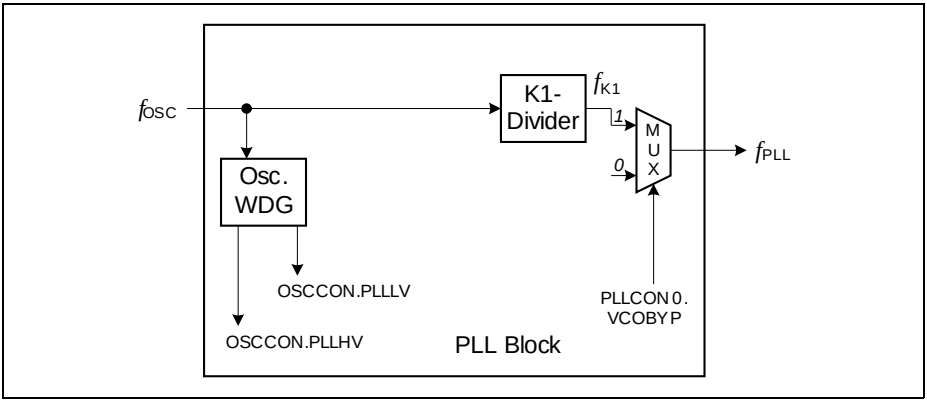


Figure 8-6 PLL Prescaler Mode Diagram

The output frequency is given by:

$$f_{PLL} = \frac{f_{OSC}}{K1} \quad (8.9)$$

The Prescaler Mode is selected by the following settings

- PLLCON0.VCOBYP = 1

The Prescaler Mode is entered when the following requirement isvalid:

- PLLSTAT.VCOBYST = 1

Operation on the Prescaler Mode does require an input clock frequency of f_{OSC}. Therefore it is recommended to check and monitor if an input frequency f_{OSC} is available at all by checking OSCCON.PLLLV. For a better monitoring also the upper frequency can be monitored via OSCCON.PLLHV.

For the Prescaler Mode there are no requirements regarding the frequency of f_{OSC}.

The system operation frequency is controlled in the Prescaler Mode by the value of the K1-Divider. When the value of PLLCON1.K1DIV was changed the next update of this value should not be done before bit PLLSTAT.K1RDY is set.

Note: Changing the system operation frequency by changing the value of the K1-Divider has a direct coupling to the power consumption of the device. Therefore this has to be done carefully.

Depending on the selected divider value of the K1-Divider the duty cycle of the clock is selected. This can have an impact for the operation with an external communication interface. The duty cycles values for the different K1-divider values are defined in the Data Sheet.

The Prescaler Mode is requested from the Freerunning or Normal Mode by setting bit PLLCON.VCOBYP. The Prescaler Mode is entered when the status bit PLLSTAT.VCOBYST is set. Before the Prescaler Mode is requested the K1-Divider should be configured with a value generating a PLL output frequency f_{PLL} that matches the one generated by the Freerunning or Normal Mode as much as possible. In this way the frequency change resulting out of the mode change is reduced to a minimum.

The Prescaler Mode is requested to be left by clearing bit PLLCON.VCOBYP. The Prescaler Mode is left when the status bit PLLSTAT.VCOBYST is cleared.

Configuration and Operation of the Normal Mode

In Normal Mode, the PLL is running at the external frequency f_{OSC} divided down by a factor P, multiplied by a factor N and then divided down by a factor K2 / K3 resulting in f_{PLL} / f_{PLL2}.

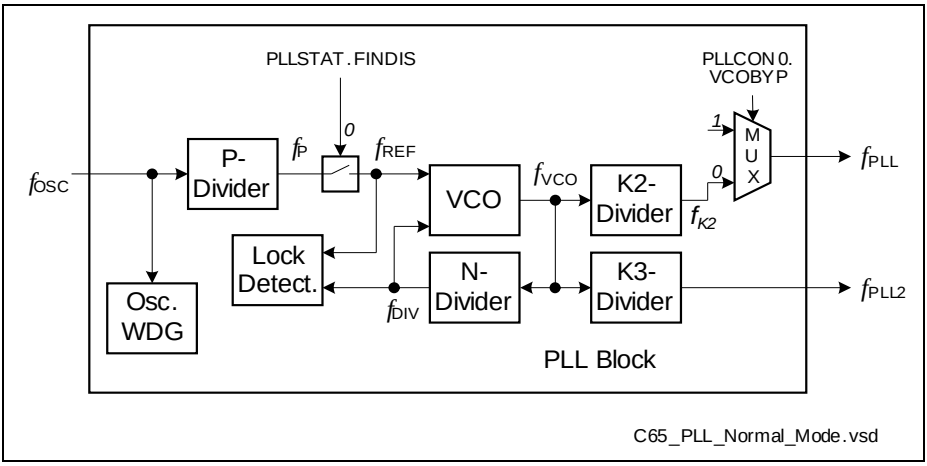


Figure 8-7 PLL Normal Mode Diagram

The output frequency is given by:

(8.10)

$$f_{PLL} = \frac{N}{P \cdot K2} \cdot f_{OSC}$$

(8.11)

$$f_{PLL2} = \frac{N}{P \cdot K3} \cdot f_{OSC}$$

The Normal Mode is selected by the following settings

- PLLCON0.VCOBYP = 0
- PLLCON0.CLRFINDIS = 1

The Normal Mode is entered when the following requirements are all together valid:

- PLLSTAT.FINDIS = 0
- PLLSTAT.VCOBYST = 0
- PLLSTAT.VCOLOCK = 1
- OSCCON.PLLLV = 1
- OSCCON.PLLHV = 1

Operation on the Normal Mode does require an input clock frequency of f_{OSC}. Therefore it is recommended to check and monitor if an input frequency f_{OSC} is available at all by checking OSCCON.PLLLV. For a better monitoring also the upper frequency can be monitored via OSCCON.PLLHV.

The system operation frequency is controlled in the Normal Mode by the values of the three dividers: P, N, and K2 / K3. A modification of the two dividers P and N has a direct influence to the VCO frequency and could lead to a loss of the VCO Lock status. A modification of the K2 / K3-divider has no impact on the VCO Lock status but still changes the PLL output frequency.

Note: Changing the system operation frequency by changing the value of the K2-Divider has a direct coupling to the power consumption of the device. Therefore this has to be done carefully.

When the frequency of the Normal Mode should be modified or entered the following sequence should be followed:

First the Prescaler Mode should be configured and entered. For more details see the Prescaler Mode.

The SMU alarm generation for the VCO Loss of Lock should be disabled.

While the Prescaler Mode is used the Normal Mode can be configured and checked for a positive VCO Lock status. The first target frequency of the Normal Mode should be selected in a way that it matches or is only slightly higher as the one used in the Prescaler Mode. This avoids big changes in the system operation frequency and therefore power consumption when switching later from Prescaler Mode to Normal Mode. The P and N divider should be selected in the following way:

- Selecting P and N in a way that f_{VCO} is in the lower area of its allowed values leads to a slightly reduced power consumption but to a slightly increased jitter
- Selecting P and N in a way that f_{VCO} is in the upper area of its allowed values leads to a slightly increased power consumption but to a slightly reduced jitter

After the P, N, and K2 / K3 dividers have been updated on the first configuration, the indication of the VCO Lock status should be checked (PLLSTAT.VCOLOCK = 1).

Note: When configuring the PLL for the first time after a system reset it is recommended to disconnect the input clock f_{OSC} before configuring P and / or N and to connect f_{OSC} before checking for the lock status.

Note: It is recommended to reset the VCO Lock detection (PLLCON0.RESLD = 1) after the new values of the dividers are configured to get a defined VCO lock check time.

Note: Resetting the lock detection (PLLCON0.RESLD = 1) while PLLCON0.OSCDISCDIS = 0 when the PLL was already locked (PLLSTAT.VCOLOCK = 1) can lead to a disconnect of the input clock. In this case PLLCON0.OSCDISCDIS should be set before lock detection is reset. When the lock is thereafter is set again PLLCON0.OSCDISCDIS could be cleared again.

When this happens the switch from Prescaler Mode to Normal Mode can be done. Normal Mode is requested by clearing PLLCON.VCOBYP. The Normal Mode is entered when the status bit PLLSTAT.VCOBYST is cleared.

Now the Normal Mode is entered. The SMU status flag for the system PLL VCO Loss-of-Lock event should be cleared and then enabled again. The intended PLL output target frequency can now be configured by changing only the K2 / K3-Divider.

Depending on the selected divider value of the K2-Divider the duty cycle of the clock is selected. This can have an impact for the operation with an external communication interface.

This can result in multiple changes of the K2-Divider to avoid too big frequency changes. Between the update of two K2-Divider values 6 cycles of f_{PLL} should be waited.

PLL VCO Lock Detection

The PLL has a lock detection that supervises the VCO part of the PLL in order to differentiate between stable and instable VCO circuit behavior. The lock detector marks the VCO circuit and therefore the output f_{VCO} of the VCO as instable if the two inputs f_{REF} and f_{DIV} differ too much. Changes in one or both input frequencies below a level are not marked by a loss of lock because the VCO can handle such small changes without any problem for the system.

PLL VCO Loss-of-Lock Event

The PLL may become unlocked, caused by a break of the crystal / ceramic resonator or the external clock line. In such a case, an SMU alarm event is generated.

Additionally, the OSC clock input f_{OSC} is disconnected from the PLL VCO to avoid unstable operation due to noise or sporadic clock pulses coming from the oscillator circuit. Without a clock input f_{OSC}, the PLL gradually slows down to its VCO base frequency and remains there. This automatic feature can be disabled by setting bit PLLCON0.OSCDISCDIS. If this bit is set the OSC clock remains connected to the VCO.

VCO Power Down Mode

The PLL offers a VCO Power Down Mode. This mode can be entered to save power within the PLL. The VCO Power Down Mode is entered by setting bit PLLCON0.VCOPWD. While the PLL is in VCO Power Down Mode only the Prescaler Mode is operable. Please note that selecting the VCO Power Down Mode does not automatically switch to the Prescaler Mode. So before the VCO Power Down Mode is entered the Prescaler Mode must be active.

PLL Power Down Mode

The PLL offers a Power Down Mode. This mode can be entered to save power if the PLL is not needed at all. The Power Down Mode is entered by setting bit PLLCON0.PLLPWD. While the PLL is in Power Down Mode no PLL output frequency is generated.

Frequency Modulation

If the PLL operates in Normal Mode the output frequency f_{PLL} can additionally be modified by a low-frequency modulation. f_{VCO} is randomly modulated.

The modulation is enabled via bit PLLCON0.MODEN. The modulation itself variate the VCO frequency randomly with the range of the configured modulation amplitude. The modulation amplitude is selected via PLLCON2.MODCFG[9:0].

(8.12)

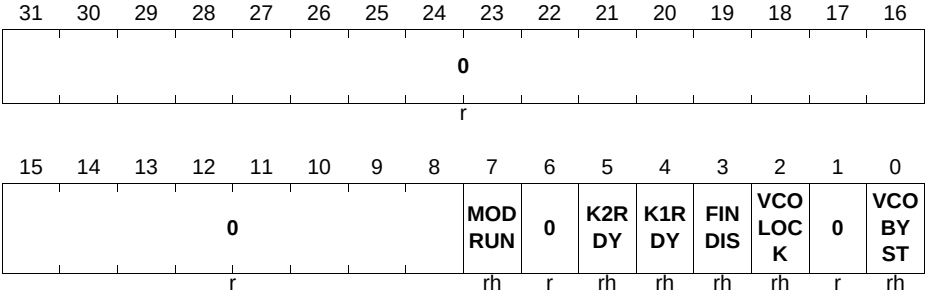
$$MA = \frac{f_{MV}}{f_{VCO} \times 2} \times MODCFG[9,0]$$

The modulation performed in a way that the resulting accumulated jitter added by the modulation stays below J_{MOD}. The modulation itself is monitored with f_{REF} and therefore the P-divider should by configured with the smallest possible value.

PLL Registers

These registers control the setting of the PLL.

PLLSTAT
PLL Status Register **(014_H)** **Reset Value: 0000 0038$_H$**



Field	Bits	Type	Description
VCOBYST	0	rh	**VCO Bypass Status** 0_B Freerunning / Normal Mode is entered 1_B Prescaler Mode is entered
VCOLOCK	2	rh	**PLL VCO Lock Status** 0_B The frequency difference of f_{REF} and f_{DIV} is greater than allowed. The VCO part of the PLL can not lock on a target frequency. 1_B The frequency difference of f_{REF} and f_{DIV} is small enough to enable a stable VCO operation. *Note: In case of a loss of VCO lock the f_{VCO} is keep on the previous constant frequency.*
FINDIS	3	rh	**Input Clock Disconnect Select Status** 0_B The input clock from the oscillator is connected to the VCO part 1_B The input clock from the oscillator is disconnected from the VCO part *Note: This bit can be set by setting bit PLLCON0.SETFINDIS.* *Note: This bit can be cleared by setting bit PLLCON0.CLRFINDIS.*

Field	Bits	Type	Description
K1RDY	4	rh	**K1 Divider Ready Status** This bit indicates if the K1-divider operates on the configured value or not. this is of interest if the values is changed. 0_B K1-Divider is not ready to operate with the new value 1_B K1-Divider is ready to operate with the new value
K2RDY	5	rh	**K2 Divider Ready Status** This bit indicates if the K2-divider operates on the configured value or not. this is of interest if the values is changed. 0_B K2-Divider is not ready to operate with the new value 1_B K2-Divider is ready to operate with the new value
MODRUN	7	rh	**Modulation Run** This bit indicates if the frequency modulation of the PLL is activated or not. 0_B Frequency modulation is not active 1_B Frequency modulation is active
0	1, 6, [31:8]	r	**Reserved** Read as 0; should be written with 0.

PLLCON0
PLL Configuration 0 Register **(018_H)** **Reset Value: 0001 C600$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				PDIV				0					RES LD	0	PLL PWD
r				rw				r					w	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDIV							0	0	OSC DISC DIS	CLR FIN DIS	SET FIN DIS	0	MOD EN	VCO PWD	VCO BYP
rw							rw	r	rw	w	w	rw	rw	rw	rw

Field	Bits	Type	Description
VCOBYP	0	rw	**VCO Bypass** 0_B Normal operation, VCO is not bypassed 1_B Prescaler Mode; VCO is bypassed
VCOPWD	1	rw	**VCO Power Saving Mode** 0_B Normal behavior 1_B The VCO is put into a Power Saving Mode and can no longer be used. Only Prescaler Mode are active if previously selected.
MODEN	2	rw	**Modulation Enable** This bit controls the activation of the frequency modulation of the PLL. 0_B Frequency modulation is not activated 1_B Frequency modulation is activated
SETFINDIS	4	w	**Set Status Bit PLLSTAT.FINDIS** 0_B Bit PLLSTAT.FINDIS is left unchanged 1_B Bit PLLSTAT.FINDIS is set. The input clock from the oscillator is disconnected from the VCO part.
CLRFINDIS	5	w	**Clear Status Bit PLLSTAT.FINDIS** 0_B Bit PLLSTAT.FINDIS is left unchanged 1_B Bit PLLSTAT.FINDIS is cleared. The input clock from the oscillator is connected to the VCO part.
OSCDISCDIS	6	rw	**Oscillator Disconnect Disable** This bit is used to disable the control PLLSTAT.FINDIS in a PLL loss-of-lock case. 0_B In case of a PLL loss-of-lock bit PLLSTAT.FINDIS is set 1_B In case of a PLL loss-of-lock bit PLLSTAT.FINDIS is cleared
NDIV	[15:9]	rw	**N-Divider Value** The value the N-Divider operates is NDIV+1.
PLLPWD	16	rw	**PLL Power Saving Mode** 0_B The complete PLL block is put into a Power Saving Mode and can no longer be used. 1_B Normal behavior

Field	Bits	Type	Description
RESLD	18	w	**Restart VCO Lock Detection** Setting this bit will clear bit PLLSTAT.VCOLOCK and restart the VCO lock detection. Reading this bit returns always a zero.
PDIV	[27:24]	rw	**P-Divider Value** The value the P-Divider operates is PDIV+1.
0	3, 8	rw	**Reserved** Have to be written with 0.
0	7, 17, [23:19], [31:28]	r	**Reserved** Read as 0; should be written with 0.

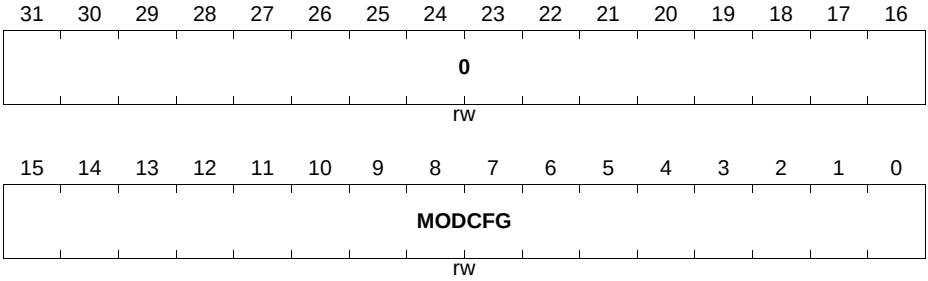
PLLCON1
PLL Configuration 1 Register **(01C$_H$)** **Reset Value: 0002 020F$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0									K1DIV						
r									rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	K3DIV							0	K2DIV						
r	rw							r	rw						

Field	Bits	Type	Description
K2DIV	[6:0]	rw	**K2-Divider Value** The value the K2-Divider operates is K2DIV+1.
K3DIV	[14:8]	rw	**K3-Divider Value** The value the K3-Divider operates is K3DIV+1.
K1DIV	[22:16]	rw	**K1-Divider Value** The value the K1-Divider operates is K1DIV+1.
0	7, 15, [31:23]	r	**Reserved** Read as 0; should be written with 0.

PLLCON2
PLL Configuration 2 Register **(020_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
MODCFG	[15:0]	rw	**Modulation Configuration** This bit field defines the modulation. MODCFG[9:0] defines the modulation amplitude. Bits MODCFG[9:5] are treated as integer part and bits MODCFG[4:0] as fractional part. Bits MODCFG[15:10] have to be configured with the following setting: $0x111101_B$.
0	[31:16]	rw	**Reserved** Have to be written with 0.

8.1.2.2 ERAY Phase-Locked Loop (PLL_ERAY) Module

The PLL_ERAY can convert a low-frequency external clock signal to a high-speed internal clock for maximum performance. It can execute emergency actions if it loses its lock on the external clock.

This module is a phase locked loop for integer frequency synthesis. It allows the use of input and output frequencies of a wide range by varying the different divider factors.

Features

- VCO lock detection
- 4-bit input divider **P:** (divide by PDIV+1)
- 5-bit feedback divider **N**: (multiply by NDIV+1)
- 7-bit output divider **K1 or K2:** (divide by either by K1DIV+1 or K2DIV+1)
- 7-bit output divider **K3:** (divide by K3DIV+1)

- Different operating modes
  - Prescaler Mode
  - Freerunning Mode
  - Normal Mode
- VCO Power Down (Sleep Mode)
- Glitchless switching between both K-Dividers
- Glitchless switching between Normal Mode and Prescaler Mode

PLL_ERAY Functional Description

The following figure shows the PLL_ERAY block structure.

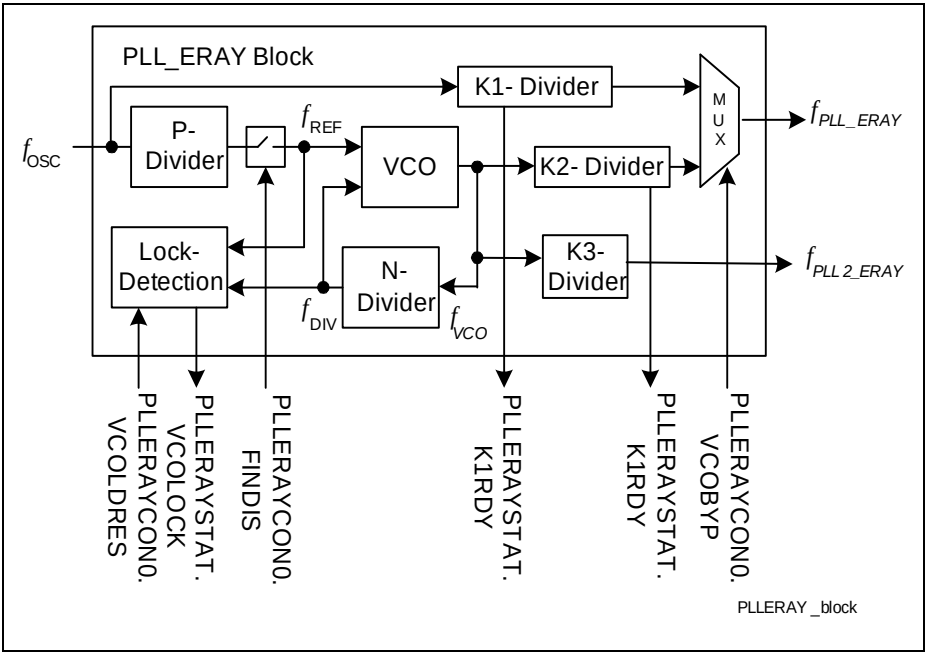


Figure 8-8 PLL_ERAY Block Diagram

Clock Source Control

The PLL_ERAY clocks f_{PLL_ERAY} / f_{PLL2_ERAY} are generated from f_{OSC} in one of three software selectable modes:

- Normal Mode
- Prescaler Mode
- Freerunning Mode

Normal Mode

In Normal Mode the input frequency f_{OSC} is divided down by a factor P, multiplied by a factor N and then divided down by a factor K2 / K3.

The output frequency is given by

$$f_{PLLERAY} = \frac{N}{P \times K2} \cdot f_{OSC} \quad (8.13)$$

$$f_{PLL2ERAY} = \frac{N}{P \times K3} \cdot f_{OSC} \quad (8.14)$$

Prescaler Mode

In Prescaler Mode the reference frequency f_{OSC} is only divided down by a factor K1.

The output frequency is given by

$$f_{PLLERAY} = \frac{f_{OSC}}{K1} \quad (8.15)$$

Freerunning Mode

In Freerunning Mode the base frequency output of the Voltage Controlled Oscillator (VCO) $f_{PLLBASEERAY}$ is only divided down by a factor K2 / K3.

The output frequency is given by

$$f_{PLLERAY} = \frac{f_{PLLBASEERAY}}{K2} \quad (8.16)$$

$$f_{PLL2ERAY} = \frac{f_{PLLBASEERAY}}{K3} \quad (8.17)$$

Configuration and Operation of the Freerunning Mode

In Freerunning Mode, the PLL_ERAY is running at its VCO base frequency and f_{PLL_ERAY} / f_{PLL2_ERAY} are derived from f_{VCO} only by the K2 / K3-Divider.

The Freerunning Mode is entered after each System Reset.

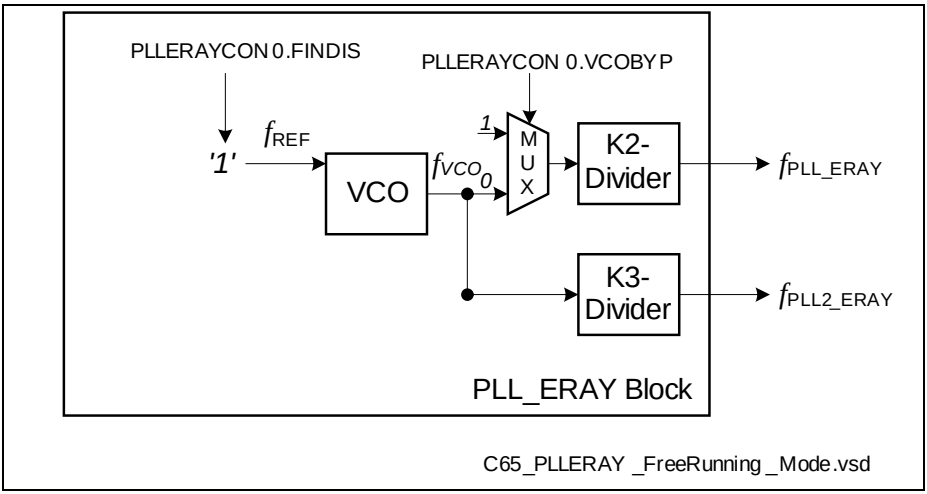


Figure 8-9 PLL_ERAY Free-Running Mode Diagram

The output frequency is given by

$$f_{PLLERAY} = \frac{f_{PLLBASEERAY}}{K2} \quad (8.18)$$

$$f_{PLL2ERAY} = \frac{f_{PLLBASEERAY}}{K3} \quad (8.19)$$

The Freerunning Mode is selected by the following settings

- PLLERAYCON0.VCOBYP = 0
- PLLERAYCON0.SETFINDIS = 1

The Freerunning Mode is entered when

- PLLERAYSTAT.FINDIS = 1
- AND
- PLLERAYSTAT.VCOBYST = 0

Operation on the Freerunning Mode does not require an input clock frequency of f_{OSC}. The Freerunning Mode is automatically entered on a PLL_ERAY VCO Loss-of-Lock event if bit PLLERAYCON0.OSCDISCDIS is cleared. This mechanism allows a fail-safe operation of the PLL_ERAY as in emergency cases still a clock is available.

The frequency of the Freerunning Mode $f_{PLLBASEERAY}$ is listed in the Data Sheet.

Depending on the selected divider value of the K2-Divider the duty cycle of the clock is selected. This can have an impact for the operation with an external communication interface.

Configuration and Operation of the Prescaler Mode

In Prescaler Mode, the PLL_ERAY is running at the external frequency f_{OSC} and f_{PLL_ERAY} is derived from f_{OSC} only by the K1-Divider.

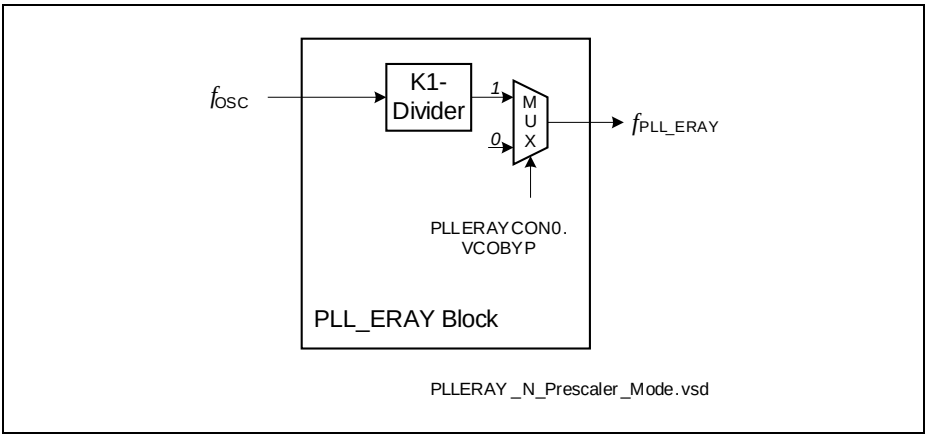


Figure 8-10 PLL_ERAY Prescaler Mode Diagram

The output frequency is given by:

$$f_{PLLERAY} = \frac{f_{OSC}}{K1} \quad (8.20)$$

The Prescaler Mode is selected by the following settings

- PLLERAYCON0.VCOBYP = 1

The Prescaler Mode is entered when the following requirement isvalid:

- PLLERAYSTAT.VCOBYST = 1

Operation on the Prescaler Mode does require an input clock frequency of f_{OSC}. Therefore it is recommended to check and monitor if an input frequency f_{OSC} is available at all by checking OSCCON.PLLLV. For a better monitoring also the upper frequency can be monitored via OSCCON.PLLHV.

For the Prescaler Mode there are no requirements regarding the frequency of f_{OSC}.

The system operation frequency is controlled in the Prescaler Mode by the value of the K1-Divider. When the value of PLLERAYCON1.K1DIV was changed the next update of this value should not be done before bit PLLERAYSTAT.K1RDY is set.

Depending on the selected divider value of the K1-Divider the duty cycle of the clock is selected. This can have an impact for the operation with an external communication interface. The duty cycles values for the different K1-divider values are defined in the Data Sheet.

The Prescaler Mode is requested from the Freerunning or Normal Mode by setting bit PLLERAYCON.VCOBYP. The Prescaler Mode is entered when the status bit PLLERAYSTAT.VCOBYST is set. Before the Prescaler Mode is requested the K1-Divider should be configured with a value generating a PLL_ERAY output frequency f_{PLL_ERAY} that matches the one generated by the Freerunning or Normal Mode as much as possible. In this way the frequency change resulting out of the mode change is reduced to a minimum.

The Prescaler Mode is requested to be left by clearing bit PLLERAYCON.VCOBYP. The Prescaler Mode is left when the status bit PLLERAYSTAT.VCOBYST is cleared.

Configuration and Operation of the Normal Mode

In Normal Mode, the PLL is running at the external frequency f_{OSC} divided down by a factor P, multiplied by a factor N and then divided down by a factor K2 / K3 resulting in $f_{PLLERAY}$ / $f_{PLL2ERAY}$.

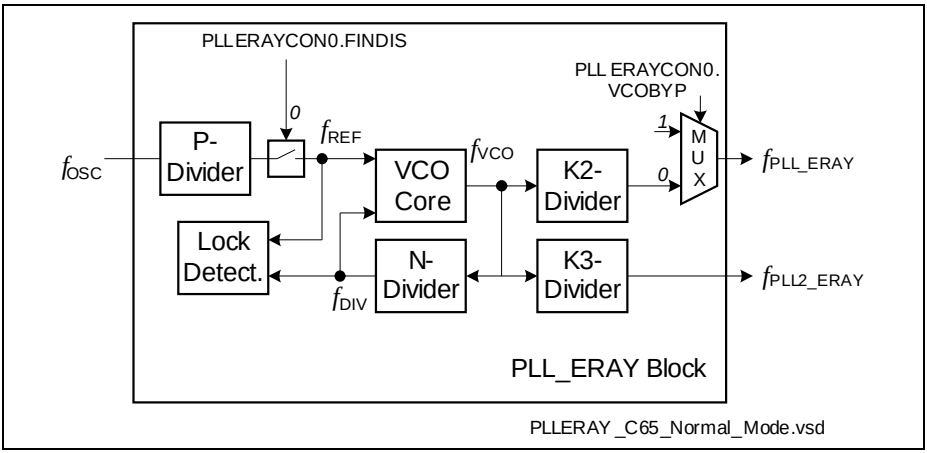


Figure 8-11 PLL_ERAY Normal Mode Diagram

The output frequency is given by:

(8.21)

$$f_{PLLERAY} = \frac{N}{P \times K2} \cdot f_{OSC}$$

(8.22)

$$f_{PLL2ERAY} = \frac{N}{P \times K3} \cdot f_{OSC}$$

The Normal Mode is selected by the following settings

- PLLERAYCON0.VCOBYP = 0
- PLLERAYCON0.CLRFINDIS = 1

The Normal Mode is entered when the following requirements are all together valid:

- PLLERAYSTAT.FINDIS = 0
- PLLERAYSTAT.VCOBYST = 0
- PLLERAYSTAT.VCOLOCK = 1
- OSCCON.PLLLV = 1
- OSCCON.PLLHV = 1

Operation on the Normal Mode does require an input clock frequency of f_{OSC}. Therefore it is recommended to check and monitor if an input frequency f_{OSC} is available at all by checking OSCCON.PLLLV. For a better monitoring also the upper frequency can be monitored via OSCCON.PLLHV.

The system operation frequency is controlled in the Normal Mode by the values of the three dividers: P, N and K2 / K3. A modification of the two dividers P and N has a direct influence to the VCO frequency and could lead to a loss of the VCO Lock status. A modification of the K2 / K3-divider has no impact on the VCO Lock status but still changes the PLL_ERAY output frequency.

When the frequency of the Normal Mode should be modified or entered the following sequence should be followed:

First the Prescaler Mode should be configured and entered. For more details see the Prescaler Mode.

The SMU alarm generation for the VCO Loss of Lock should be disabled.

While the Prescaler Mode is used the Normal Mode can be configured and checked for a positive VCO Lock status. The first target frequency of the Normal Mode should be selected in a way that it matches or is only slightly higher as the one used in the Prescaler Mode. This avoids big changes in the system operation frequency and therefore power consumption when switching later from Prescaler Mode to Normal Mode. The N divider should be selected in the following way:

- Selecting P and N in a way that f_{VCO} is in the lower area of its allowed values leads to a slightly reduced power consumption but to a slightly increased jitter
- Selecting P and N in a way that f_{VCO} is in the upper area of its allowed values leads to a slightly increased power consumption but to a slightly reduced jitter

After the P, N and K2 / K3 dividers are updated for the first configuration the indication of the VCO Lock status should be await (PLLERAYSTAT.VCOLOCK = 1).

Note: When configuring the PLL the input clock f_{OSC} should be disconnected before configuring P and / or N and connected before checking for the lock status.

Note: It is recommended to reset the VCO Lock detection (PLLERAYCON0.RESLD = 1) after the new values of the dividers are configured to get a defined VCO lock check time.

Note: Resetting the lock detection (PLLERAYCON0.RESLD = 1) while PLLERAYCON0.OSCDISCDIS = 0 when the PLL_ERAY was already locked (PLLERAYSTAT.VCOLOCK = 1) can lead to a disconnect of the input clock. In this case PLLERAYCON0.OSCDISCDIS should be set before lock detection is reset. If the lock is subsequently set again PLLERAYCON0.OSCDISCDIS could be cleared again.

When this happens the switch from Prescaler Mode to Normal Mode can be done. Normal Mode is requested by clearing PLLERAYCON.VCOBYP. The Normal Mode is entered when the status bit PLLERAYSTAT.VCOBYST is cleared.

Now the Normal Mode is entered. The SMU alarm flag for the PLL_ERAY VCO Loss-of-Lock event should be cleared and then enabled again. The intended PLL_ERAY output target frequency can not be configured by changing only the K2-Divider.

Depending on the selected divider value of the K2-Divider the duty cycle of the clock is selected. This can have an impact for the operation with an external communication interface. This can result in multiple changes of the K2-Divider to avoid too big frequency changes. Between the update of two K2-Divider values 6 cycles of f_{PLL_ERAY} should be waited.

PLL_ERAY VCO Lock Detection

The PLL_ERAY has a lock detection that supervises the VCO part of the PLL_ERAY in order to differentiate between stable and instable VCO circuit behavior. The lock detector marks the VCO circuit and therefore the output f_{VCO} of the VCO as instable if the two inputs f_{REF} and f_{DIV} differ too much. Changes in one or both input frequencies below a level are not marked by a loss of lock because the VCO can handle such small changes without any problem for the system.

PLL_ERAY VCO Loss-of-Lock Event

The PLL_ERAY may become unlocked, caused by a break of the crystal / ceramic resonator or the external clock line. In such a case, an SMU alarm event is generated.

Additionally, the OSC clock input f_{OSC} is disconnected from the PLL_ERAY VCO to avoid unstable operation due to noise or sporadic clock pulses coming from the oscillator circuit. Without a clock input f_{OSC}, the PLL_ERAY gradually slows down to its VCO base frequency and remains there. This automatic feature can be disabled by setting bit PLLERAYCON0.OSCDISCDIS. If this bit is cleared the OSC clock remains connected to the VCO.

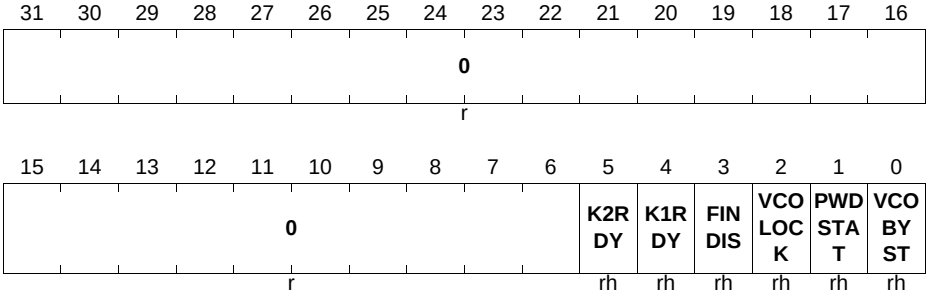
VCO Power Down Mode

The PLL_ERAY offers a VCO Power Down Mode. This mode can be entered to save power within the PLL_ERAY. The VCO Power Down Mode is entered by setting bit PLLCON0.VCOPWD. While the PLL_ERAY is in VCO Power Down Mode only the Prescaler Mode is operable. Please note that selecting the VCO Power Down Mode does not automatically switch to the Prescaler Mode. So before the VCO Power Down Mode is entered, the Prescaler Mode must be active.

PLL_ERAY Registers

These registers controls the setting of the PLL_ERAY.

PLLERAYSTAT
PLL_ERAY Status Register **(024_H)** **Reset Value: 0000 0038$_H$**



Field	Bits	Type	Description
VCOBYST	0	rh	**VCO Bypass Status** 0_B Freerunning / Normal Mode is entered 1_B Prescaler Mode is entered
PWDSTAT	1	rh	**PLL_ERAY Power-saving Mode Status** 0_B PLL_ERAY Power-saving Mode was not entered 1_B PLL_ERAY Power-saving Mode was entered
VCOLOCK	2	rh	**PLL VCO Lock Status** 0_B The frequency difference of f_{REF} and f_{DIV} is greater than allowed. The VCO part of the PLL_ERAY can not lock on a target frequency. 1_B The frequency difference of f_{REF} and f_{DIV} is small enough to enable a stable VCO operation. *Note: In case of a loss of VCO lock the f_{VCO} goes to the upper boundary of the VCO frequency if the reference clock input is greater than expected.* *Note: In case of a loss of VCO lock the f_{VCO} goes to the lower boundary of the VCO frequency if the reference clock input is lower than expected.*

Field	Bits	Type	Description
FINDIS	3	rh	**Input Clock Disconnect Select Status** 0_B The input clock from the oscillator is connected to the VCO part 1_B The input clock from the oscillator is disconnected from the VCO part *Note: This bit can be set by setting bit PLLERAYCON0.SETFINDIS.* *Note: This bit can be cleared by setting bit PLLERAYCON0.CLRFINDIS.*
K1RDY	4	rh	**K1 Divider Ready Status** This bit indicates if the K1-divider operates on the configured value or not. this is of interest if the values is changed. 0_B K1-Divider is not ready to operate with the new value 1_B K1-Divider is ready to operate with the new value
K2RDY	5	rh	**K2 Divider Ready Status** This bit indicates if the K2-divider operates on the configured value or not. this is of interest if the values is changed. 0_B K2-Divider is not ready to operate with the new value 1_B K2-Divider is ready to operate with the new value
0	[31:6]	r	**Reserved** Read as 0; should be written with 0.

PLLERAYCON0

PLL_ERAY Configuration 0 Register (028$_H$) **Reset Value: 0001 2E00$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				PDIV				0					RES LD	0	PLL PWD
r				rw				r					w	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	NDIV					0		OSC DISC DIS	CLR FIN DIS	SET FIN DIS	0		VCO PWD	VCO BYP
r	rw	rw					r		rw	w	w	r		rw	rw

Field	Bits	Type	Description
VCOBYP	0	rw	**VCO Bypass** 0_B Normal operation, VCO is not bypassed 1_B Prescaler Mode; VCO is bypassed
VCOPWD	1	rw	**VCO Power Saving Mode** 0_B Normal behavior 1_B The VCO is put into a Power Saving Mode and can no longer be used.
SETFINDIS	4	w	**Set Status Bit PLLERAYSTAT.FINDIS** 0_B Bit PLLERAYSTAT.FINDIS is left unchanged 1_B Bit PLLERAYSTAT.FINDIS is set. The input clock from the oscillator is disconnected from the VCO part.
CLRFINDIS	5	w	**Clear Status Bit PLLERAYSTAT.FINDIS** 0_B Bit PLLERAYSTAT.FINDIS is left unchanged 1_B Bit PLLERAYSTAT.FINDIS is cleared. The input clock from the oscillator is connected to the VCO part.
OSCDISCDIS	6	rw	**Oscillator Disconnect Disable** This bit is used to disable the control PLLERAYSTAT.FINDIS in a PLL_ERAY loss-of-lock case. 0_B In case of a PLL loss-of-lock bit PLLERAYSTAT.FINDIS is set 1_B In case of a PLL loss-of-lock bit PLLERAYSTAT.FINDIS is cleared

Field	Bits	Type	Description
NDIV	[13:9]	rw	**N-Divider Value** The value the N-Divider operates is (NDIV+1).
PLLPWD	16	rw	**PLL Power Saving Mode** 0_B The complete PLL_ERAY block is put into a Power Saving Mode and can no longer be used. Only the Bypass Mode is active if previously selected. 1_B Normal behavior
RESLD	18	w	**Restart VCO Lock Detection** Setting this bit will clear bit PLLERAYSTAT.VCOLOCK and restart the VCO lock detection. Reading this bit returns always a zero.
PDIV	[27:24]	rw	**P-Divider Value** The value the P-Divider operates is PDIV+1.
0	14	rw	**Reserved** Should be written with 0.
0	[3:2], [8:7], 15, 17, [23:19],[31:28]	r	**Reserved** Read as 0; should be written with 0.

PLLERAYCON1
PLL_ERAY Configuration 1 Register (02C$_H$) **Reset Value: 000F 020F$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0									K1DIV						
r									rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				K3DIV				0	K2DIV						
r				rw				r	rw						

Field	Bits	Type	Description
K2DIV	[6:0]	rw	**K2-Divider Value** The value the K2-Divider operates is K2DIV+1.
K3DIV	[11:8]	rw	**K3-Divider Value** The value the K1-Divider operates is K3DIV+1.
K1DIV	[22:16]	rw	**K1-Divider Value** The value the K1-Divider operates is K1DIV+1.
0	7, [15:12], [31:23]	r	**Reserved** Read as 0; should be written with 0.

8.1.3 Clock Distribution

Using the first two parts of the Clock System all clocks the system rely on for operation are defined. Now these different clocks need to be distributed to the single modules, CPU, and blocks in a way that these IPs can operate in best way in terms performance and power consumption.

For the clock distribution the system is split into several sub-clock domains where the clock speed could be configured individually but there are also several limitation for each sub-clock domain derived out of the internal interfaces.

Each sub-clock domain defines a logical unit from clocking perspective point of view.

The clock distribution is done via the Clock Control Unit (CCU). The CCU receives the clocks that are created by the two PLLs ($f_{PLL/PLL2}$ and f_{PLL_ERAY/PLL_ERAY2}), the back-up clock f_{Back}, and f_{OSC0}. These clock are either forwarded directly or divided in order to supply the sub-clock domains.

The following figure shows the structure of the TC21x/TC22x/TC23x clock system that shows the different sub-clock domains together with their clock inputs.

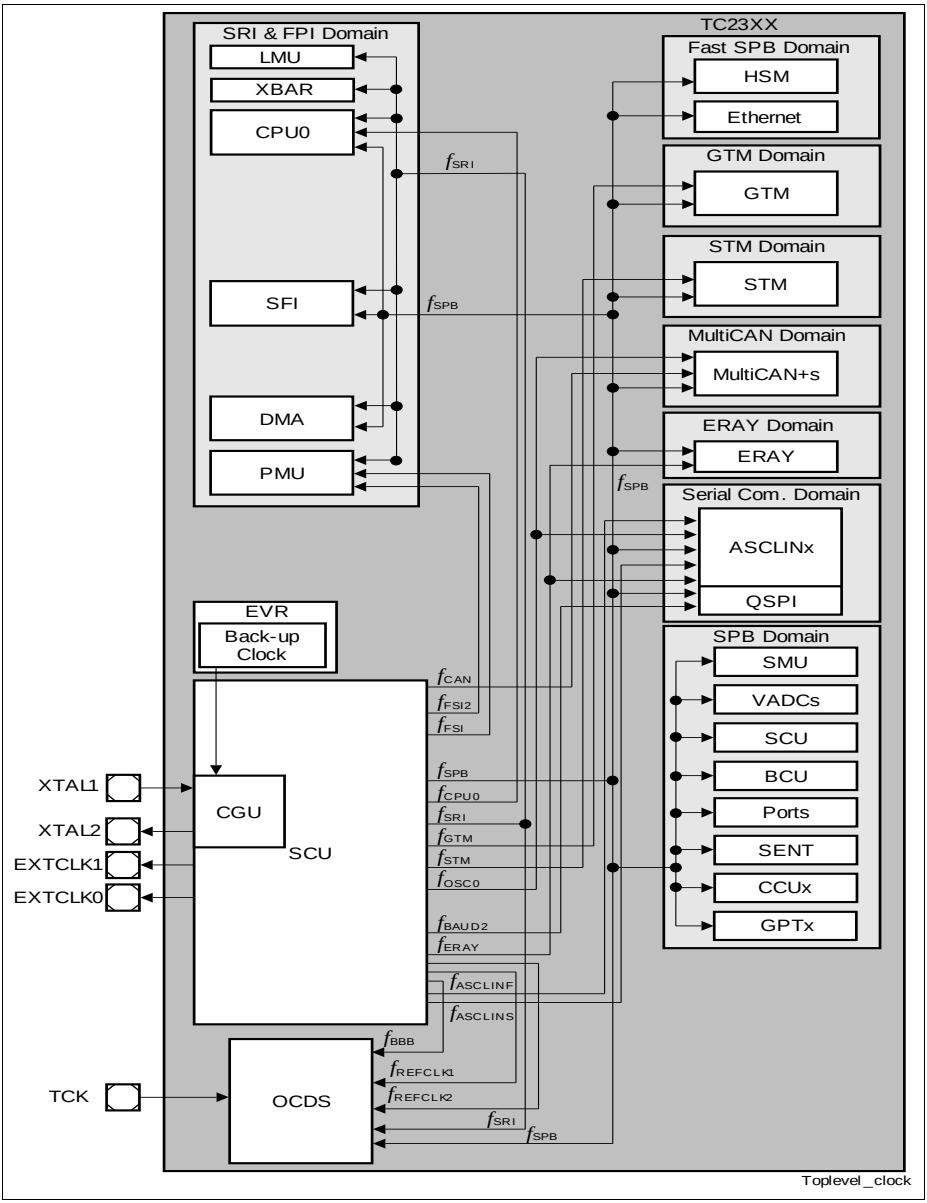


Figure 8-12 TC21x/TC22x/TC23x Clocking System

8.1.3.1 Clock Control Unit

The Clock Control Unit (CCU) receives the clocks that are created by the two PLLs ($f_{PLL/PLL2}$ and $f_{PLL_ERAY/PLL2_ERAY}$), the back-up clock f_{Back}, and f_{OSC0}.

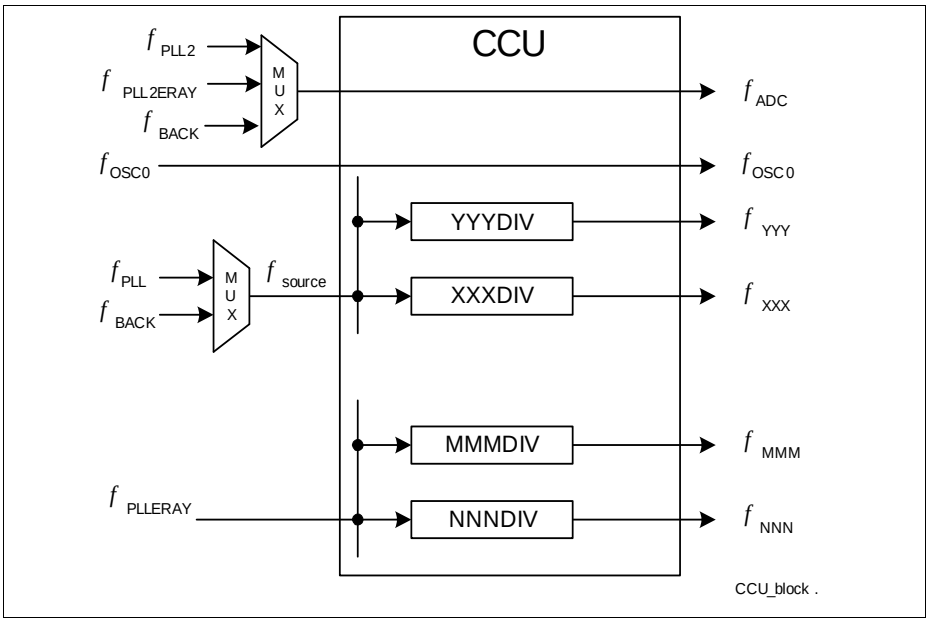


Figure 8-13 Clock Control Unit Overview

For most clocks a linear divider is provided to adapt the clock frequency to the application requirement. This divider is controlled by the bit field XXXDIV.

For the CPU clocks, more sophisticated dividers are implemented. This allows a better precision for the control in clock frequency changes of individual CPUs. This mechanism allows the adaptation of the frequency change to a specific current limit defined by the application.

There is also a fixed reference clock REFCLK1/2 for the debug block, which divides the master clock f_{PLL}/$f_{PLLERAY}$ by 24. This allows the OCDS to generate timestamps independently of the selected SRI and SPB clock speeds.

Basic Clock System Mechanisms

The clocking system offers a lot of options and a high amount of flexibility. Behind this complex clocking system a set of basic ideas is implemented in order to allow a configuration as close as possible to the given application needs.

The heart of the system is a synchronous block including all modules except the ERAY. The heartbeat here is defined by f_{MAX}. f_{MAX} is defined as the highest frequency used inside the system for any clock derived out of f_{SOURCE}.

For different IPs (CPUs, modules or module clusters) options are included to influence the application execution / performance via the clock control.

Following is a brief overview of the different clocks:

- System buses
  - f_{SRI} defines the operating performance of the SRI-Bus and therefore the data exchange rate between all connected masters and slaves
  - f_{SPB} defines the operating performance of the SPB-Bus and therefore the data exchange rate between all connected masters and slaves and the interrupt system
- CPU controls
  - f_{CPU0} defines the execution speed of CPU0
- PMU controls
  - f_{FSI2} defines the execution speed of the PFlash for reading operations
  - f_{FSI} defines the execution speed for all other flash operations
- Peripheral options
  - f_{STM} defines a basic frequency for the STMs independent of the rest of the system (beside the limitations listed in **Table 8-2**). This allows the STMs to operate on a constant frequency.
  - f_{GTM} defines a basic frequency for the GTM independent of the rest of the system (beside the limitations listed in **Table 8-2**). This allows the GTM to operate on a constant frequency.
  - f_{CAN} defines a basic frequency for the MultiCAN independent of the rest of the system (beside the limitations listed in **Table 8-2**). This allows the MultiCAN to operate on a constant baudrate (frequency).
  - $f_{ASCLINS}$ defines a basic frequency for the ASCLINs independent of the rest of the system (beside the limitations listed in **Table 8-2**). This allows the ASCLINs to operate on a constant baudrate (frequency).
- Debug system
  - As debugging should be non-intrusive to the application system, a separate clock f_{BBB} for dedicated debug resources is available. This allows debugging (trace generation) during changes of the other clock configurations. Please note that f_{BBB} needs to be faster than or equal to f_{SPB} for debug.

In addition several peripherals offer the option to operate on other clock sources. If frequency modulation is used, some peripherals should switch to a non-modulated clock.

Due to its unique requirements the ERAY module is implemented as an asynchronous domain with its own independent clock source.

Table 8-1 CCU Clock Options

CCU Clock Output	Clock Source			
	PLL	PLL_ERAY	Back-up	OSC_XTAL
f_{MAX}	✓	–	Default	–
f_{SRI}	✓	–	Default	–
f_{CPU0}	f_{SRI}	–	–	–
f_{SPB}	✓	–	Default	–
f_{FSI}	f_{SRI}	–	–	–
f_{FSI2}	f_{SRI}	–	–	–
$f_{REFCLK1/2}$	✓	✓	Default	–
f_{BBB}	✓	–	Default	–
f_{ERAY}	–	Default	–	–
f_{GTM}	✓	–	Default	–
f_{STM}	✓	–	Default	–
f_{BAUD2}	✓	–	Default	–
f_{CAN}	✓	–	Default	–
$f_{ASCLINF}$	✓	–	Default	–
$f_{ASCLINS}$	✓	–	Default	–

The complete system is based on a single clock (f_{SOURCE}) in order to achieve the performance advantage of a synchronous system. Therefore certain limitations have to be considered when configuring the clock control options.

Clock Divider Limitations

For the clock dividers which control the different sub-clock domains / modules the allowed values are limited and the following ratios have to be observed. The ratios are defined in the following way: clock A = f_{AAA}; clock B = f_{BBB} the allowed ratio is 1 : n where n has a defined range of values. Clock A is always faster or equal to clock B in this definition with f_{AAA} [MHz] = n * f_{BBB} [MHz].

In addition the divider values are limited by the resulting maximum frequencies . These allowed max. frequencies are defined in the Target Data Sheet / ACDC Target Specification.

Table 8-2 CCU allowed Clock Ratios

Clock A	Clock B	Allowed Ratios [1]	Recommended Default
f_{SRI}	f_{SPB}	1 : n; n = 1, 2, 3, 4, 5, 6	n = 2
f_{SRI}	f_{FSI}	1 : n; n = 1, 2	n = 2
f_{SRI}	f_{FSI2}	1 : n; n = 1, 2	n = 1
f_{FSI2}	f_{FSI}	1 : n; n = 1, 2	n = 2
f_{GTM}	f_{SPB}	1 : n [2]; n = 1, 2, 3, 4, 5,...	n = 1
f_{SPB}	f_{GTM}	1 : n [2]; n = 1, 2, 3, 4, 5,...	n = 1
f_{SPB}	f_{STM}	1 : n [3]; n = 1, 2, 3, 4, 5,...	n = 1
f_{STM}	f_{SPB}	1 : n [3]; n = 1, 2, 3, 4, 5,...	n = 1
f_{SRI}	f_{BBB} [4]	1 : n; n = 1, 2	n = 2
f_{BAUD2}	f_{SPB}	1 : n [5]; n = 1, 2, 3, 4, 5,...	n = 2
f_{SPB}	f_{BAUD2}	1 : n [5]; n = 1, 2, 3, 4, 5,...	-
f_{CAN}	f_{SPB}	1 : n [6]; n = 1, 2, 3, 4, 5,...	n = 1
f_{SPB}	f_{CAN}	1 : n [6]; n = 1, 2, 3, 4, 5,...	-
$f_{ASCLINF}$	f_{SPB}	1 : n [7]; n = 1, 2, 3, 4, 5,...	n = 2
f_{SPB}	$f_{ASCLINF}$	1 : n [7]; n = 1, 2, 3, 4, 5,...	-
$f_{ASCLINS}$	f_{SPB}	1 : n [8]; n = 1, 2, 3, 4, 5,...	n = 1
f_{SPB}	$f_{ASCLINS}$	1 : n[8]; n = 1, 2, 3, 4, 5,...	-

1) Configuring the registers CCUCON0, CCUCON1 and CCUCON2 to values that result in a violation of the allowed ratios are not allowed even if a single configuration entery for a bit field is with the allowed register bit field description.

2) f_{GTM} can be faster, slower, or equal to f_{SPB}

3) f_{STM} can be faster, slower, or equal to f_{SPB}

4) f_{BBB} has to be faster, or equal to f_{SPB}

5) f_{BAUD2} can be faster, slower, or equal to f_{SPB}

6) f_{CAN} can be faster, slower, or equal to f_{SPB}

7) $f_{ASCLINF}$ can be faster, slower, or equal to f_{SPB}

8) $f_{ASCLINS}$ can be faster, slower, or equal to f_{SPB}

Note: Recommended values did not necessarily reflect the default configuration after a reset event. Instead they should give a hint how the configure the system for a optimal performance configuration.

CCU Registers

*Note: The relation in between the described CCUCONx.yyyDIV bit fields is described in the table **CCU allowed Clock Ratios**.*

CCUCON0
CCU Clock Control Register 0 (030_H) Reset Value: 0112 0148_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LCK	**UP**	**CLKSEL**		**0**		**FSIDIV**		**0**		**FSI2DIV**		**SPBDIV**			
rh	w	rw		rw		rw		rw		rw		rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPDIV				**SRIDIV**				**BAUD2DIV**				**1**	**0**		
rw				rw				rw				rw	rw		

Field	Bits	Type	Function
BAUD2DIV	[7:4]	rw	**Baud2 Divider Reload Value** The resulting Baud2 frequency is configured to $f_{BAUD2} = f_{source}$ / BAUD2DIV for the allowed configurations. For BAUD2DIV = 0000_B the clock is shut off. f_{source} could be configured either to f_{PLL} (CLKSEL = 01_B) or f_{BACK} (CLKSEL = 00_B) 0000_B f_{BAUD2} is stopped 0001_B $f_{BAUD2} = f_{source}$ 0010_B $f_{BAUD2} = f_{source}/2$ 0011_B $f_{BAUD2} = f_{source}/3$ 0100_B $f_{BAUD2} = f_{source}/4$ 0101_B $f_{BAUD2} = f_{source}/5$ 0110_B $f_{BAUD2} = f_{source}/6$ 0111_BReserved, do not use this combination 1000_B $f_{BAUD2} = f_{source}/8$ 1001_BReserved, do not use this combination 1010_B $f_{BAUD2} = f_{source}/10$ 1011_BReserved, do not use this combination 1100_B $f_{BAUD2} = f_{source}/12$ 1101_BReserved, do not use this combination 1110_BReserved, do not use this combination 1111_B $f_{BAUD2} = f_{source}/15$

Field	Bits	Type	Function
SRIDIV	[11:8]	rw	**SRI Divider Reload Value** The resulting SRI frequency is configured to $f_{SRI} = f_{source}$ / SRIDIV for the allowed configurations. For SRIDIV = 0000_B the clock is shut off. f_{source} could be configured either to f_{PLL} (CLKSEL = 01_B) or f_{BACK} (CLKSEL = 00_B) 0000_B f_{SRI} is stopped 0001_B $f_{SRI} = f_{source}$ 0010_B $f_{SRI} = f_{source}/2$ 0011_B $f_{SRI} = f_{source}/3$ 0100_B $f_{SRI} = f_{source}/4$ 0101_B $f_{SRI} = f_{source}/5$ 0110_B $f_{SRI} = f_{source}/6$ 0111_BReserved, do not use this combination 1000_B $f_{SRI} = f_{source}/8$ 1001_BReserved, do not use this combination 1010_B $f_{SRI} = f_{source}/10$ 1011_BReserved, do not use this combination 1100_B $f_{SRI} = f_{source}/12$ 1101_BReserved, do not use this combination 1110_BReserved, do not use this combination 1111_B $f_{SRI} = f_{source}/15$
LPDIV	[15:12]	rw	**Low Power Divider Reload Value** $0000_B$$f_{MAX}$, f_{SRI}, f_{SPB}, and f_{BBB} are controlled by the dedicated CCUCONx bit fields $0001_B$$f_{SRI}$, f_{SPB}, and $f_{BBB} = f_{source}/30$; $f_{MAX} = f_{source}/15$ $0010_B$$f_{SRI}$, f_{SPB}, and $f_{BBB} = f_{source}/60$; $f_{MAX} = f_{source}/30$ $0011_B$$f_{SRI}$, f_{SPB}, and $f_{BBB} = f_{source}/120$; $f_{MAX} = f_{source}/60$ $0100_B$$f_{SRI}$, f_{SPB}, and $f_{BBB} = f_{source}/240$; $f_{MAX} = f_{source}/120$ 0101_BReserved, do not use this combination 0110_BReserved, do not use this combination 0111_BReserved, do not use this combination 1000_BReserved, do not use this combination 1001_BReserved, do not use this combination 1010_BReserved, do not use this combination 1011_BReserved, do not use this combination 1100_BReserved, do not use this combination 1101_BReserved, do not use this combination 1110_BReserved, do not use this combination 1111_BReserved, do not use this combination

Field	Bits	Type	Function
SPBDIV	[19:16]	rw	**SPB Divider Reload Value** 0000_B Reserved, do not use this combination 0001_B Reserved, do not use this combination 0010_B $f_{SPB} = f_{source}/2$ 0011_B $f_{SPB} = f_{source}/3$ 0100_B $f_{SPB} = f_{source}/4$ 0101_B $f_{SPB} = f_{source}/5$ 0110_B $f_{SPB} = f_{source}/6$ 0111_B Reserved, do not use this combination; resulting in $f_{SPB} = f_{source}/6$ 1000_B $f_{SPB} = f_{source}/8$ 1001_B Reserved, do not use this combination; resulting in $f_{SPB} = f_{source}/8$ 1010_B $f_{SPB} = f_{source}/10$ 1011_B Reserved, do not use this combination; resulting in $f_{SPB} = f_{source}/10$ 1100_B $f_{SPB} = f_{source}/12$ 1101_B Reserved, do not use this combination; resulting in $f_{SPB} = f_{source}/12$ 1110_B Reserved, do not use this combination; resulting in $f_{SPB} = f_{source}/12$ 1111_B $f_{SPB} = f_{source}/15$ f_{source} could be configured either to f_{PLL} ($CLKSEL = 01_B$) or f_{BACK} ($CLKSEL = 00_B$)
FSI2DIV	[21:20]	rw	**FSI2 Divider Reload Value** 00_B f_{FSI2} is shut off 01_B $f_{FSI2} = f_{SRI}$ 10_B $f_{FSI2} = f_{SRI}$ / 2 for SRIDIV = 0001_B or 0010_B, else $f_{FSI2} = f_{SRI}$ 11_B $f_{FSI2} = f_{SRI}$ / 3 for SRIDIV = 0001_B or 0010_B, else $f_{FSI2} = f_{SRI}$
FSIDIV	[25:24]	rw	**FSI Divider Reload Value** 00_B f_{FSI} is shut off 01_B $f_{FSI} = f_{SRI}$ 10_B $f_{FSI} = f_{SRI}$ / 2 for SRIDIV = 0001_B or 0010_B, else $f_{FSI} = f_{SRI}$ 11_B $f_{FSI} = f_{SRI}$ / 3 for SRIDIV = 0001_B or 0010_B, else $f_{FSI} = f_{SRI}$

Field	Bits	Type	Function
CLKSEL	[29:28]	rw	**Clock Selection** This bit field defines the clock source that is used for the clock generation of f_{SOURCE}. 00_B back-up clock is used as clock source f_{source} 01_B f_{PLL} is used as clock source f_{source} 10_B Reserved, do not use this combination 11_B Reserved, do not use this combination
UP	30	w	**Update Request** This bit defined the update for the CCU. Please note that setting this bit request a CCU update based on all three registers, so only one UP bit have to be set. 0_B No action 1_B A new complete parameter set is transferred to the CCU. All three registers CCUCON0, 1 and 5 content is taken by CCU. This bit always read as zero.
LCK	31	rh	**Lock Status** This bit indicates if the register can be updated with a new value or if the register is locked and a write action from the bus side has no effect. 0_B The register is unlocked and can be updated 1_B The register is locked and can not be updated
1	3	rw	**Reserved** Should be written with 1.
0	[2:0], [23:22] ,[27:26]	rw	**Reserved** Should be written with 0.

*Note: The relation in between the described CCUCONx.yyyDIV bit fields is described in the table **CCU allowed Clock Ratios**.*

CCUCON1
CCU Clock Control Register 1

(034_H) **Reset Value: 0000 2211$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LCK	**UP**	**INSEL**		**ASCLINSDIV**				**ASCLINFDIV**				**ETHDIV**			
rh	w	rw		rw				rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GTMDIV				**STMDIV**				**ERAYDIV**				**CANDIV**			
rw				rw				rw				rw			

Field	Bits	Type	Function
CANDIV	[3:0]	rw	**MultiCAN Divider Reload Value** The resulting MultiCAN frequency is configured to $f_{CAN} = f_{source}$ / CANDIV for the allowed configurations. For CANDIV = 0000_B the clock is shut off. f_{source} could be configured either to f_{PLL} (CLKSEL = 01_B) or f_{BACK} (CLKSEL = 00_B) 0000_B f_{CAN} is stopped 0001_B $f_{CAN} = f_{source}$ 0010_B $f_{CAN} = f_{source}/2$ 0011_B $f_{CAN} = f_{source}/3$ 0100_B $f_{CAN} = f_{source}/4$ 0101_B $f_{CAN} = f_{source}/5$ 0110_B $f_{CAN} = f_{source}/6$ 0111_BReserved, do not use this combination 1000_B $f_{CAN} = f_{source}/8$ 1001_BReserved, do not use this combination 1010_B $f_{CAN} = f_{source}/10$ 1011_BReserved, do not use this combination; 1100_B $f_{CAN} = f_{source}/12$ 1101_BReserved, do not use this combination 1110_BReserved, do not use this combination 1111_B $f_{CAN} = f_{source}/15$

Field	Bits	Type	Function
ERAYDIV	[7:4]	rw	**ERAY Divider Reload Value** The resulting ERAY frequency is configured to $f_{ERAY} = f_{PLLERAY}$ / ERAYDIV for the allowed configurations. For ERAYDIV = 0000_B the clock is shut off. 0000_B f_{ERAY} is stopped 0001_B $f_{ERAY} = f_{PLLERAY}$ 0010_B $f_{ERAY} = f_{PLLERAY}/2$ 0011_B $f_{ERAY} = f_{PLLERAY}/3$ 0100_B $f_{ERAY} = f_{PLLERAY}/4$ 0101_B $f_{ERAY} = f_{PLLERAY}/5$ 0110_B $f_{ERAY} = f_{PLLERAY}/6$ 0111_BReserved, do not use this combination 1000_B $f_{ERAY} = f_{PLLERAY}/8$ 1001_BReserved, do not use this combination 1010_B $f_{ERAY} = f_{PLLERAY}/10$ 1011_BReserved, do not use this combination 1100_B $f_{ERAY} = f_{PLLERAY}/12$ 1101_BReserved, do not use this combination 1110_BReserved, do not use this combination 1111_B $f_{ERAY} = f_{PLLERAY}/15$ This bit field is not link to the update handshake defined by bits UP and LCK. An update of this bit field has no influence to the other clocks in the system. Always a new value (different number) is written to this bit field an update is done. If the same value is written again no update is started.

Field	Bits	Type	Function
STMDIV	[11:8]	rw	**STM Divider Reload Value** The resulting STM frequency is configured to $f_{STM} = f_{source}$ / STMDIV for the allowed configurations. For STMDIV = 0000_B the clock is shut off. f_{source} could be configured either to f_{PLL} ($CLKSEL = 01_B$) or f_{BACK} ($CLKSEL = 00_B$) 0000_B f_{STM} is stopped 0001_B $f_{STM} = f_{source}$ 0010_B $f_{STM} = f_{source}/2$ 0011_B $f_{STM} = f_{source}/3$ 0100_B $f_{STM} = f_{source}/4$ 0101_B $f_{STM} = f_{source}/5$ 0110_B $f_{STM} = f_{source}/6$ 0111_BReserved, do not use this combination 1000_B $f_{STM} = f_{source}/8$ 1001_BReserved, do not use this combination 1010_B $f_{STM} = f_{source}/10$ 1011_BReserved, do not use this combination 1100_B $f_{STM} = f_{source}/12$ 1101_BReserved, do not use this combination 1110_BReserved, do not use this combination 1111_B $f_{STM} = f_{source}/15$

Field	Bits	Type	Function
GTMDIV	[15:12]	rw	**GTM Divider Reload Value** The resulting GTM frequency is configured to $f_{GTM} = f_{source}$ / GTMDIV for the allowed configurations. For GTMDIV = 0000_B the clock is shut off. f_{source} could be configured either to f_{PLL} (CLKSEL = 01_B) or f_{BACK} (CLKSEL = 00_B) 0000_B f_{GTM} is stopped 0001_B $f_{GTM} = f_{source}$ 0010_B $f_{GTM} = f_{source}/2$ 0011_B $f_{GTM} = f_{source}/3$ 0100_B $f_{GTM} = f_{source}/4$ 0101_B $f_{GTM} = f_{source}/5$ 0110_B $f_{GTM} = f_{source}/6$ 0111_BReserved, do not use this combination 1000_B $f_{GTM} = f_{source}/8$ 1001_BReserved, do not use this combination 1010_B $f_{GTM} = f_{source}/10$ 1011_BReserved, do not use this combination 1100_B $f_{GTM} = f_{source}/12$ 1101_BReserved, do not use this combination 1110_BReserved, do not use this combination 1111_B $f_{GTM} = f_{source}/15$

Field	Bits	Type	Function
ETHDIV	[19:16]	rw	**Ethernet Divider Reload Value** The resulting Ethernet frequency is configured to $f_{ETH} = f_{PLLERAY}$ / ETHDIV * 4 for the allowed configurations. For ETHDIV = 0000_B the clock is shut off. 0000_B f_{ETH} is stopped 0001_B $f_{ETH} = f_{PLLERAY/4}$ 0010_B $f_{ETH} = f_{PLLERAY}/8$ 0011_B $f_{ETH} = f_{PLLERAY}/12$ 0100_B $f_{ETH} = f_{PLLERAY}/16$ 0101_B $f_{ETH} = f_{PLLERAY}/20$ 0110_B $f_{ETH} = f_{PLLERAY}/24$ 0111_BReserved, do not use this combination 1000_B $f_{ETH} = f_{PLLERAY}/32$ 1001_BReserved, do not use this combination 1010_B $f_{ETH} = f_{PLLERAY}/40$ 1011_BReserved, do not use this combination 1100_B $f_{ETH} = f_{PLLERAY}/48$ 1101_BReserved, do not use this combination 1110_BReserved, do not use this combination 1111_B $f_{ETH} = f_{PLLERAY}/60$ This bit field is not link to the update handshake defined by bits UP and LCK. An update of this bit field has no influence to the other clocks in the system. Always a new value (different number) is written to this bit field an update is done. If the same value is written again no update is started. A 50% duty cycle clock is generated for f_{ETH}.

Field	Bits	Type	Function
ASCLINFDIV	[23:20]	rw	**ASCLIN Fast Divider Reload Value** The resulting ASCLINF frequency is configured to $f_{ASCLINF} = f_{source}$ / ASCLINFDIV for the allowed configurations. For ASCLINFDIV = 0000_B the clock is shut off. f_{source} could be configured either to f_{PLL} (CLKSEL = 01_B) or f_{BACK} (CLKSEL = 00_B) 0000_B $f_{ASCLINF}$ is stopped 0001_B $f_{ASCLINF} = f_{source}$ 0010_B $f_{ASCLINF} = f_{source}/2$ 0011_B $f_{ASCLINF} = f_{source}/3$ 0100_B $f_{ASCLINF} = f_{source}/4$ 0101_B $f_{ASCLINF} = f_{source}/5$ 0110_B $f_{ASCLINF} = f_{source}/6$ 0111_BReserved, do not use this combination 1000_B $f_{ASCLINF} = f_{source}/8$ 1001_BReserved, do not use this combination 1010_B $f_{ASCLINF} = f_{source}/10$ 1011_BReserved, do not use this combination 1100_B $f_{ASCLINF} = f_{source}/12$ 1101_BReserved, do not use this combination 1110_BReserved, do not use this combination 1111_B $f_{ASCLINF} = f_{source}/15$

Field	Bits	Type	Function
ASCLINSDIV	[27:24]	rw	**ASCLIN Slow Divider Reload Value** The resulting ASCLINS frequency is configured to $f_{ASCLINS} = f_{source}$ / ASCLINSDIV for the allowed configurations. For ASCLINSDIV = 0000_B the clock is shut off. f_{source} could be configured either to f_{PLL} (CLKSEL = 01_B) or f_{BACK} (CLKSEL = 00_B) 0000_B $f_{ASCLINS}$ is stopped 0001_B $f_{ASCLINS} = f_{source}$ 0010_B $f_{ASCLINS} = f_{source}/2$ 0011_B $f_{ASCLINS} = f_{source}/3$ 0100_B $f_{ASCLINS} = f_{source}/4$ 0101_B $f_{ASCLINS} = f_{source}/5$ 0110_B $f_{ASCLINS} = f_{source}/6$ 0111_BReserved, do not use this combination 1000_B $f_{ASCLINS} = f_{source}/8$ 1001_BReserved, do not use this combination 1010_B $f_{ASCLINS} = f_{source}/10$ 1011_BReserved, do not use this combination 1100_B $f_{ASCLINS} = f_{source}/12$ 1101_BReserved, do not use this combination 1110_BReserved, do not use this combination 1111_B $f_{ASCLINS} = f_{source}/15$
INSEL	[29:28]	rw	**Input Selection** This bit field defines as clock source for the two PLLs (PLL and PLL_ERAY). 00_B back-up clock is used as clock source 01_B f_{OSC0} is used as clock source 10_B Reserved, do not use this combination 11_B Reserved, do not use this combination
UP	30	w	**Update Request** This bit defined the update for the CCU. Please note that setting this bit request a CCU update based on all three registers, so only one UP bit have to be set. 0_B No action 1_B A new complete parameter set is transferred to the CCU. All three registers CCUCON0, 1 and 5 content is taken by CCU. This bit always read as zero.

Field	Bits	Type	Function
LCK	31	rh	**Lock Status** This bit indicates if the register can be updated with a new value or if the register is locked and a write action from the bus side has no effect. 0_B The register is unlocked and can be updated 1_B The register is locked and can not be updated

CCUCON5
CCU Clock Control Register 5

($04C_H$) **Reset Value: 0000 0041$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LCK	**UP**	**0**													
rh	w	r													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0									**1**	**0**		**MAXDIV**			
rw									rw	rw		rw			

Field	Bits	Type	Function
MAXDIV	[3:0]	rw	**Max Divider Reload Value** The resulting frequency is configured to $f_{MAX} = f_{source}$ / MAXDIV. For MAXDIV = 0000_B $f_{MAX} = f_{source}$. f_{source} could be configured either to f_{PLL} (CCUCON0.CLKSEL = 01_B) or f_{BACK} (CCUCON0.CLKSEL = 00_B) 0000_B $f_{MAX} = f_{source}$ 0001_B $f_{MAX} = f_{source}$ 0010_B $f_{MAX} = f_{source}/2$ 0011_B $f_{MAX} = f_{source}/3$ 0100_B $f_{MAX} = f_{source}/4$ 0101_B $f_{MAX} = f_{source}/5$ 0110_B $f_{MAX} = f_{source}/6$ 0111_BReserved, do not use this combination 1000_B $f_{MAX} = f_{source}/8$ 1001_BReserved, do not use this combination 1010_B $f_{MAX} = f_{source}/10$ 1011_BReserved, do not use this combination 1100_B $f_{MAX} = f_{source}/12$ 1101_BReserved, do not use this combination 1110_BReserved, do not use this combination 1111_B $f_{MAX} = f_{source}/15$ *Note: For power saving it's recommended to configure MAXDIV always with the same value as CCUCON0.SRIDIV. Otherwise MAXDIV could be set always to 0001_B.*
1	6	rw	**Reserved** Should be written with 1.
0	[5:4], [15:7]	rw	**Reserved** Should be written with 0.
0	[29:16]	r	**Reserved** Read as 0; should be written with 0.

Field	Bits	Type	Function
UP	30	w	**Update Request** This bit defined the update for the CCU. Please note that setting this bit request a CCU update based on all three registers, so only one UP bit have to be set. 0_B No action 1_B A new complete parameter set is transferred to the CCU. All three registers CCUCON0, 1 and 5 content is taken by CCU. This bit always read as zero.
LCK	31	rh	**Lock Status** This bit indicates if the register can be updated with a new value or if the register is locked and a write action from the bus side has no effect. 0_B The register is unlocked and can be updated 1_B The register is locked and can not be updated

*Note: The relation in between the described CCUCONx.yyyDIV bit fields is described in the table **CCU allowed Clock Ratios**.*

CCUCON2
CCU Clock Control Register 2

(040_H) **Reset Value: 0000 0002$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LCK	**UP**	**0**													
rh	w	r													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												**BBBDIV**			
rw												rw			

Field	Bits	Type	Function
BBBDIV	[3:0]	rw	**BBB Divider Reload Value** The resulting BBB frequency is configured to $f_{BBB} = f_{source}$ / BBBDIV for all allowed configurations. For BBBDIV = 0000_B the clock is shut off. f_{source} could be configured either to f_{PLL} (CLKSEL = 01_B) or f_{BACK} (CLKSEL = 00_B) 0000_B f_{BBB} is stopped 0001_B $f_{BBB} = f_{source}$ 0010_B $f_{BBB} = f_{source}/2$ 0011_B $f_{BBB} = f_{source}/3$ 0100_B $f_{BBB} = f_{source}/4$ 0101_B $f_{BBB} = f_{source}/5$ 0110_B $f_{BBB} = f_{source}/6$ 0111_BReserved, do not use this combination; resulting in $f_{BBB} = f_{source}/6$ 1000_B $f_{BBB} = f_{source}/8$ 1001_BReserved, do not use this combination; resulting in $f_{BBB} = f_{source}/8$ 1010_B $f_{BBB} = f_{source}/10$ 1011_BReserved, do not use this combination; resulting in $f_{BBB} = f_{source}/10$ 1100_B $f_{BBB} = f_{source}/12$ 1101_BReserved, do not use this combination; resulting in $f_{BBB} = f_{source}/12$ 1110_BReserved, do not use this combination; resulting in $f_{BBB} = f_{source}/12$ 1111_B $f_{BBB} = f_{source}/15$
UP	30	w	**Update Request** This bit defined the update for the CCU. Please note that setting this bit request a CCU update based on this register. 0_B No action 1_B A new complete parameter set is transferred to the CCU. This bit always read as zero.

Field	Bits	Type	Function
LCK	31	rh	**Lock Status** This bit indicates if the register can be updated with a new value or if the register is locked and a write action from the bus side has no effect. 0_B The register is unlocked and can be updated 1_B The register is locked and can not be updated
0	[15:4]	rw	**Reserved** Should be written with 0.
0	[29:16]	r	**Reserved** Read as 0; should be written with 0.

CCUCON6
CCU Clock Control Register 6

(080_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										**CPU0DIV**					
r										rw					

Field	Bits	Type	Function
CPU0DIV	[5:0]	rw	**CPU0 Divider Reload Value** The resulting CPU0 frequency (performance) is configured to $f_{CPU0} = f_{SRI} * (64 - CPU0DIV) / 64$. For CPU0DIV = 000000_B $f_{CPU0} = f_{SRI}$.
0	[31:6]	r	**Reserved** Read as 0; should be written with 0.

8.1.4 Individual Clock Generation

Each module provides some individual clock control options. Most modules provide a Clock Control (CLC) Register for this purpose. Some modules provide in addition more options to control the clocks. For more details please see the dedicated module chapters.

Modules can have in addition other registers controlling sub-clock domains e.g. for baud rate generation. These registers are always defined in the module chapters.

8.1.4.1 Clock Control Register CLC

All CLC registers have basically the same bit and bit field layout. However, not all CLC register functions are implemented for each peripheral module. **Table 8-3** defines in detail which bits and bit fields of the CLC registers are implemented for each clock control register.

The following functions for the module are associated with the CLC register:

- Peripheral clock static on/off control
- Module clock behavior in Sleep Mode

Module Enable/Disable Control

If a module is not used at all by an application, it can be completely shut off by setting bit DISR in its CLC register. For peripheral modules with a run mode clock divider field RMC, a second option to completely switch off the module is to set bit field RMC to 00_H. This also disables the module's operation.

The status bit DISS always indicates whether a module is currently switched off (DISS = 1) or switched on (DISS = 0).

Write operations to the non CLC registers of disabled modules are not allowed. However, the CLC of a disabled module can be written. An attempt to write to any of the other writable registers of a disabled module except CLC will cause the corresponding Bus Control Unit (BCU) to generate a bus error.

A read operation of registers of a disabled module is allowed and does not generate a bus error.

When a disabled module is switched on by writing an appropriate value to its MOD_CLC register (DISR = 0 and RMC (if implemented) > 0), status bit DISS changes from 1 to 0. During the phase in which the module becomes active, any write access to corresponding module registers (while DISS is still set) will generate a bus error. When enabling a disabled module, application software should read back the CLC register once, to check that DISS is cleared, before writing to any module register (including the CLC register).

Sleep Mode Control

The EDIS bit in the CLC register controls whether or not a module is stopped during Sleep Mode. If EDIS is 0, a Sleep Mode request can be recognized by the module and, when received, its clock is shut off.

If EDIS is set to 1, a Sleep Mode request is disregarded by the module and the module continues its operation.

Entering Disabled Mode

Software can request that a peripheral unit be put into Disabled Mode by setting DISR. A module will also be put into Disabled Mode if the Sleep Mode is requested and the module is configured to allow Sleep Mode.

In Secure Shut-off Mode, a module first finishes any operation in progress, then proceeds with an orderly shut down. When all sub-components of the module are ready to be shut down, the module's clock control unit turns off the clock. The status bit DISS is updated by the peripheral unit accordingly.

Module Clock Divider Control

Peripheral modules of the TC21x/TC22x/TC23x can have a RMC control bit field in their CLC registers. This Run Mode Clock control bit field makes it possible to slow down the CLC clock via a programmable clock divider circuit.

A value of 00_H in RMC disables the clock signals to these modules (CLC clock is switched off). If RMC is not equal to 00_H, the clock for a module register (f_{CLC}) accesses is generated as

(8.23)

$$f_{CLC} = \frac{f_{SPB}}{RMC}$$

where RMC is the content of its CLC register RMC field with a range of 1 to 255. If RMC is not available in a CLC register, the CLC clock frequency f_{CLC} is always equal to the frequency of f_{SPB}.

Note: The number of module clock cycles (wait states) that are required for a "destructive read" access (means: flags/bits are set/cleared by a read access) to a module register of a peripheral unit depends on the selected CLC clock frequency.
Therefore, a slower CLC clock (selected via bit field RMC in the CLC register) may result in a longer read cycle access time on the SPB for peripheral units with "destructive read" access.

Module Clock Register Implementations

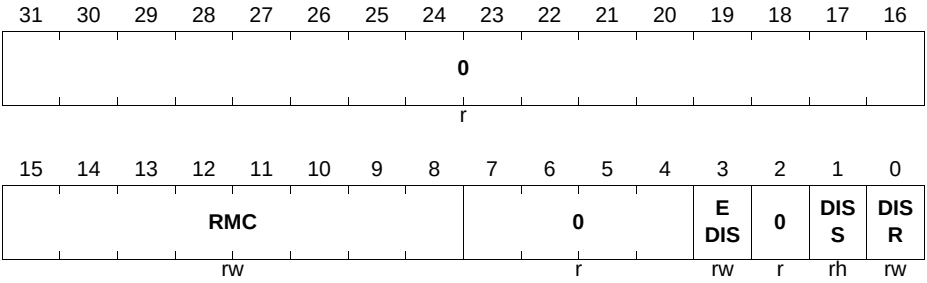
Table 8-3 shows which of the CLC register bits/bit fields are implemented for each peripheral module in the TC21x/TC22x/TC23x.

Table 8-3 Clock Generation Implementation of the TC21x/TC22x/TC23x Peripheral Modules

Module	DISR Bit 0	DISS Bit 1	EDIS Bit 3	RMC
VADC	✓	✓	✓	–
ASCLINx	✓	✓	✓	–
QSPIx	✓	✓	✓	–
GPT12x	✓	✓	✓	–
MultiCAN	✓	✓	✓	–
DMA	✓	✓	✓	–
STMx	✓	✓	✓	–
ERAY	✓	✓	✓	8-bit
LMU	✓	✓	–	–
GTM	✓	✓	–	–
CCU6x	✓	✓	✓	–
SENT	✓	✓	✓	–
CAN	✓	✓	✓	–
Ethernet	✓	✓	–	–

Module CLC Register

MOD_CLC
Clock Control Register **(00_H)** **Reset Value: Module-specific**



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. 0_B Module disable is not requested 1_B Module disable is requested
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module 0_B Module is enabled 1_B Module is disabled If the RMC field is implemented and if it is 0, DISS is set automatically.
EDIS	3	rw	**Sleep Mode Enable Control** Used for module Sleep Mode control. 0_B Sleep Mode request is regarded. Module is enabled to go into Sleep Mode. 1_B Sleep Mode request is disregarded: Sleep Mode cannot be entered on a request.
RMC	[15:8]	rw	**8-Bit Clock Divider Value in RUN Mode** This is a maximum 8-bit divider value for clock f_{SPB}. If RMC is set to 0 the module is disabled.
0	2, [7:4], [31:16]	r	**Reserved** Read as 0; should be written with 0.

8.1.5 Clock Monitors

For safety reasons clock monitors are available. For the following safety relevant clocks in the system monitors are present:

f_{PLL}
f_{PLL_ERAY}
f_{SRI}
f_{SPB}
f_{GTM}
f_{STM}

Each of these clocks is monitored by its own counter.

As reference clock the back-up clock is used as diverse clock source.

The divider has to be set in a way that the monitored frequency is one of the possible frequencies of **Table 8-4**.

Operating the Clock Monitors

The clock monitors are implemented in a way that safety system requirements are supported for a flexible system configuration of the clock system. So it is possible to adapt the monitor configuration to the application configuration of the clock system.

For each clock monitor two bit fields are available, either in register CCUCON3 or CCUCON4 depending on the monitor.

For each monitor a divider bit field and a select bit field are available for the configuration.

The CCUCONy.SELXXX bit filed defines one target frequency out of four predefined options.

Bit field CCUCONy.DIVXXX is used to divide the real clock frequency down to the selected target frequency.

Example: if the SPB is configured to operate at 100 MHz and needs to be monitored, CCUCON3.SBPDIV should be written with 0x14 and CCUCON3.SBPSEL with 00_B. This selects a target monitoring frequency of 5 MHz and divides the SPB clock by 20 resulting also in 5 MHz. If the SPB is configured to operate at 20 MHz and needs to be monitored, CCUCON3.SBPDIV should be written with 0x04 and CCUCON3.SBPSEL with 00_B.

The four target frequencies are selected in a way that all normally used frequency in a system can be covered. This means that frequencies like 66 MHz, 120 MHz, 130 MHz, 133 MHz, ... can be monitored.

Goal of the clock monitoring is to detect and signal clock malfunctions before these errors lead to not longer working system. Therefore the back-up clock is trimmed automatically and operated on f_{BACKT}. This setup guarantees that error are detected before the system could not react anymore

Table 8-4 Target trimmed Check limits

Target Frequency	LOWER value	UPPER value	SELXXX	Error can be detected for min. deviation	Error is detected for min. deviation
7.5 MHz	0x24	0x27	11_B	-1.23% +1.56%	-8.56% +9.44%
$6.\overline{6}$ MHz	0x20	0x23	10_B	-0.9% +2.38%	-8.59% +11.11%
6 MHz	0x1C	0x1F	01_B	-3.23% +1.56%	-11.13% +10.11%
5 MHz	0x17	0x1A	00_B	-3.9% +2.83%	-12.41% +12.11%

An error is detected if the reference counter has an overflow and the monitor counter of the dedicated clock is either below the lower limit (clock too slow) or greater as the upper limit (clock too fast). Errors are signaled to the SMU where the further reaction can be configured.

In addition there is a fast upper limit checking for the SPB frequency available. This limit checking checks only for high SPB compared to the maximum allowed value. A violation generates a trigger.

An error is detected when the reference counter generates an overflow AND:

- Counter_value - UPPER > 0 for the upper limit
- OR
- LOWER - Counter_value > 0 for the upper limit

Note: This feature is supported by the Infineon safety driver [safTlib] and there is no additional customer software required.

8.1.5.1 Clock Monitor Registers

These registers can be accessed by all CPUs in the system. Anyway it is suggested that only one CPU is used to control the clocks. As CPU0 is the active and available CPU after each reset this is the best choice.

CCUCON3
CCU Clock Control Register 3

(044_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LCK	UP	SLCK	0					SRISEL		SRIDIV					
rh	w	rw	r					rw		rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLLERAYSEL		PLLERAYDIV						PLLSEL		PLLDIV					
rw		rw						rw		rw					

Field	Bits	Type	Function
PLLDIV	[5:0]	rw	**PLL Divider Value** The resulting monitoring frequency is configured to $f_{PLLMON} = f_{PLL}$ / PLLDIV. For PLLDIV = 0000_B the monitoring is disabled.
PLLSEL	[7:6]	rw	**PLL Target Monitoring Frequency Selection** 00_B 5 MHz is selected as target monitoring frequency 01_B 6 MHz is selected as target monitoring frequency 10_B 6.6 MHz is selected as target monitoring frequency 11_B 7.5 MHz is selected as target monitoring frequency
PLLERAYDIV	[13:8]	rw	**PLL_ERAY Divider Value** The resulting monitoring frequency is configured to $f_{PLLERAYMON} = f_{PLLERAY}$ / PLLERAYDIV. For PLLERAYDIV = 0000_B the monitoring is disabled.

Field	Bits	Type	Function
PLLERAYSEL	[15:14]	rw	**PLL_ERAY Target Monitoring Frequency Selection** 00_B 5 MHz is selected as target monitoring frequency 01_B 6 MHz is selected as target monitoring frequency 10_B 6.$\overline{6}$ MHz is selected as target monitoring frequency 11_B 7.5 MHz is selected as target monitoring frequency This bit field is not link to the update handshake defined by bits UP and LCK. Always a new value (different number) is written to this bit field an update is done. If the same value is written again no update is started.
SRIDIV	[21:16]	rw	**SRI Divider Value** The resulting monitoring frequency is configured to $f_{SRIMON} = f_{SRI}$ / SRIDIV. For SRIDIV = 0000_B the monitoring is disabled.
SRISEL	[23:22]	rw	**SRI Target Monitoring Frequency Selection** 00_B 5 MHz is selected as target monitoring frequency 01_B 6 MHz is selected as target monitoring frequency 10_B 6.$\overline{6}$ MHz is selected as target monitoring frequency 11_B 7.5 MHz is selected as target monitoring frequency
UP	30	w	**Update Request** This bit triggers the update for the CCU, beside CCUCON3.PLLERAY. Please note that setting this bit request a CCU update based on both registers (CCUCON3 and CCUCON4), so only one UP bit have to be set. 0_B No action 1_B A new complete parameter set is transferred to the CCU. Both registers CCUCON3 and 4 content is taken by CCU. This bit always read as zero.

Field	Bits	Type	Function
LCK	31	rh	**Lock Status** This bit indicates if the register can be updated with a new value or if the register is locked and a write action from the bus side has no effect. 0_B The register is unlocked and can be updated 1_B The register is locked and can not be updated
0	[28:24]	r	**Reserved** Read as 0; should be written with 0.

CCUCON4
CCU Clock Control Register 4

(048_H) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LCK	**UP**	**SLCK**	**0**					**STMSEL**		**STMDIV**					
rh	w	rw	r					rw		rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GTMSEL		**GTMDIV**						**SPBSEL**		**SPBDIV**					
rw		rw						rw		rw					

Field	Bits	Type	Function
SPBDIV	[5:0]	rw	**SPB Divider Value** The resulting monitoring frequency is configured to $f_{SPBMON} = f_{SPB}$ / SPBDIV. For SPBDIV = 0000_B the monitoring is disabled.
SPBSEL	[7:6]	rw	**SPB Target Monitoring Frequency Selection** 00_B 5 MHz is selected as target monitoring frequency 01_B 6 MHz is selected as target monitoring frequency 10_B 6.$\overline{6}$ MHz is selected as target monitoring frequency 11_B 7.5 MHz is selected as target monitoring frequency

Field	Bits	Type	Function
GTMDIV	[13:8]	rw	**GTM Divider Value** The resulting monitoring frequency is configured to $f_{GTMMON} = f_{GTM}$ / GTMDIV. For GTMDIV = 0000_B the monitoring is disabled.
GTMSEL	[15:14]	rw	**GTM Target Monitoring Frequency Selection** 00_B 5 MHz is selected as target monitoring frequency 01_B 6 MHz is selected as target monitoring frequency 10_B 6.$\overline{6}$ MHz is selected as target monitoring frequency 11_B 7.5 MHz is selected as target monitoring frequency
STMDIV	[21:16]	rw	**STM Divider Value** The resulting monitoring frequency is configured to $f_{STMMON} = f_{STM}$ / STMDIV. For STMDIV = 0000_B the monitoring is disabled.
STMSEL	[23:22]	rw	**STM Target Monitoring Frequency Selection** 00_B 5 MHz is selected as target monitoring frequency 01_B 6 MHz is selected as target monitoring frequency 10_B 6.$\overline{6}$ MHz is selected as target monitoring frequency 11_B 7.5 MHz is selected as target monitoring frequency
UP	30	w	**Update Request** This bit triggers the update for the CCU, beside CCUCON3.PLLERAY. Please note that setting this bit request a CCU update based on both registers (CCUCON3 and CCUCON4), so only one UP bit have to be set. 0_B No action 1_B A new complete parameter set is transferred to the CCU. Both registers CCUCON3 and 4 content is taken by CCU. This bit always read as zero.

Field	Bits	Type	Function
LCK	31	rh	**Lock Status** This bit indicates if the register can be updated with a new value or if the register is locked and a write action from the bus side has no effect. 0_B The register is unlocked and can be updated 1_B The register is locked and can not be updated
0	[28:24]	r	**Reserved** Read as 0; should be written with 0.

8.1.6 Clock Emergency Behavior

In case of a clock error the CCU switches to the back-up clock f_{BACK} as clock source. The only exception here is the ERAY clock f_{ERAY} that stays operating on the PLL_ERAY. A clock error is defined by a loss of lock event of the PLL (not PLL_ERAY) while operating in Normal Mode or by an error of the PLL clock monitor.

*Note: After a clock emergency bit field CCUCON0.CLKSEL has to be re-written following a reconfigfuration of the clock system (see also **Page 8-79**) when the root cause for the clock error disappears.*

*Note: After a clock emergency bit field PLLCON0.SETFINDIS / PLLERAYCON0.SETFINDIS has to be set following by PLLCON0.CLRFINDIS / PLLERAYCON0.CLRFINDIS clear before a reconfiguration of the clock system (see also **Page 8-79**) when the root cause for the clock error disappears.*

8.1.7 External Clock Output

Two external clock outputs are provided via pins EXTCLK0 and $\overline{\text{EXTCLK1}}$. These external clocks can be enabled/disabled via bits EXTCON.EN0 for EXTCLK0 and EXTCON.EN1 for EXTCLK1. Each of the clocks that defines a clock domain can individually be selected to be seen at pins EXTCLK0 or EXTCLK1, this is configured via bit field EXTCON.SEL0/1. Changing the content of bit field EXTCON.SEL0/1 can lead to spikes at pins EXTCLK0/1.

8.1.7.1 Programmable Frequency Output for EXTCLK0

This section describes the external clock generation using the Fractional Divider.

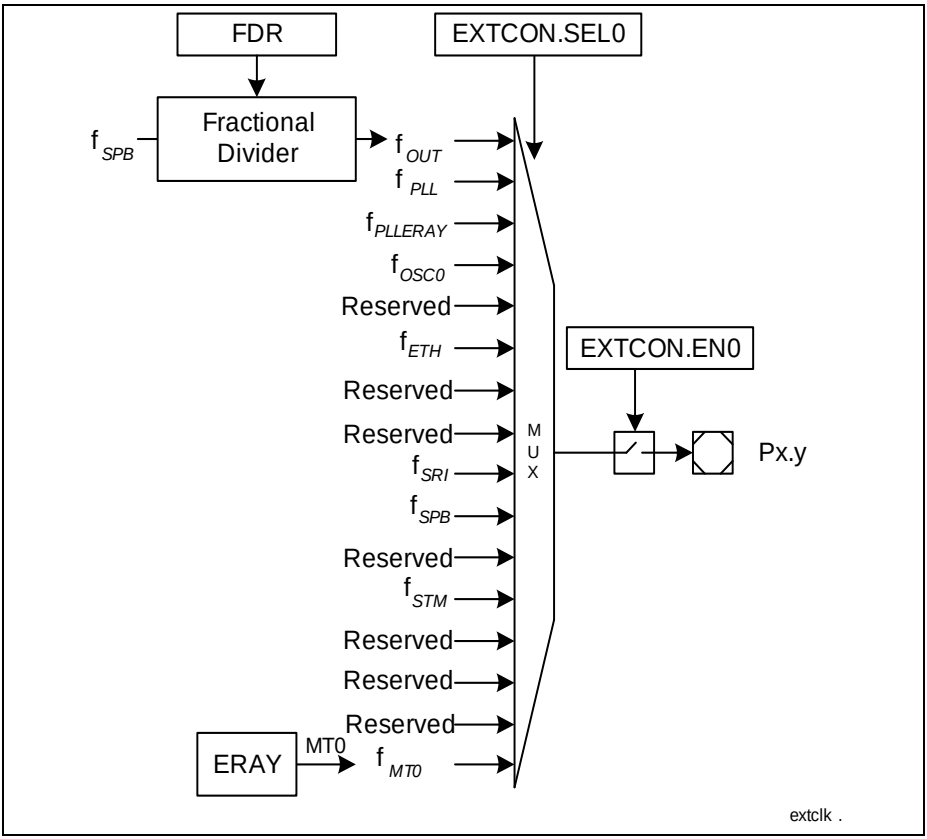


Figure 8-14 EXTCLK0 Generation

Overview

The fractional divider makes it possible to generate a external clock from the SPB clock using a programmable divider. The fractional divider divides the input clock f_{SPB} either by the factor 1/n or by a fraction of n/1024 for any value of n from 0 to 1023. This clock is thereafter divided additionally by a factor of two to guarantee a 50% duty cycle and outputs the clock, f_{OUT}. The fractional divider is controlled by the FDR register. **Figure 8-15** shows the fractional divider block diagram.

The adder logic of the fractional divider can be configured for two operating modes:

- Reload counter (addition of +1), generating an output clock pulse on counter overflow
- Adder that adds a STEP value to the RESULT value and generates an output clock pulse on counter overflow

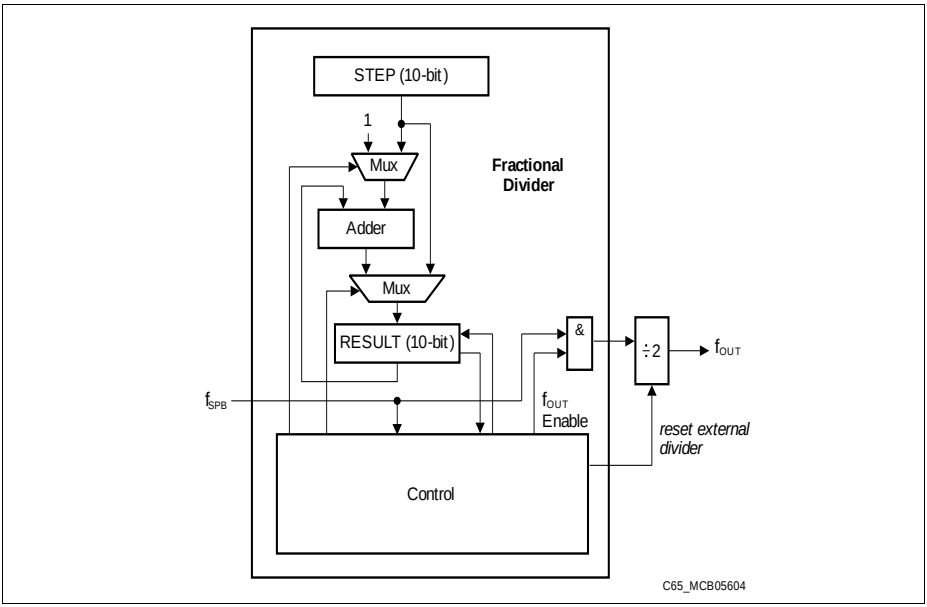


Figure 8-15 Fractional Divider Block Diagram

The adder logic of the fractional divider can be configured for two operating modes:

- **Normal Mode:** Reload counter (RESULT = RESULT + 1), generating an output clock pulse on counter overflow.
- **Fractional Divider Mode:** Adder that adds a STEP value to the RESULT value and generates an output clock pulse on counter overflow.

Fractional Divider Operating Modes

The fractional divider has two operating modes:

- Normal Divider Mode
- Fractional Divider Mode

Normal Divider Mode

In Normal Divider Mode (FDR.DM = 01_B), the fractional divider behaves as a reload counter (addition of +1) that generates an output clock pulse on the transition from $3FF_H$ to 000_H. FDR.RESULT represents the counter value and FDR.STEP determines the reload value.

The output frequencies in Normal Divider Mode are defined according to the following formulas:

$$f_{OUT} = \frac{f_{SPB} \times \frac{1}{n}}{2}, \text{ with } n = 1024 - STEP \quad (8.24)$$

In order to get $f_{OUT} = f_{SPB}/2$ STEP must be programmed with $3FF_H$.

Fractional Divider Mode

When the Fractional Divider Mode is selected (FDR.DM = 10_B), the output is derived from the input clock f_{SPB} by division of a fraction of n/1024 for any value of n from 0 to 1023 followed by the division of two. In general, the Fractional Divider Mode makes it possible to program the average output clock frequency with a higher accuracy than in Normal Divider Mode.

In Fractional Divider Mode, a pulse is generated depending on the result of the addition FDR.RESULT + FDR.STEP. If the addition leads to an overflow over $3FF_H$, a pulse is generated for the divider by two. Note that in Fractional Divider Mode the clock f_{OUT} can have a maximum period jitter of one f_{SPB} clock period.

The output frequencies in Fractional Divider Mode are defined according to the following formulas:

$$f_{OUT} = \frac{f_{SPB} \times \frac{n}{1024}}{2}, \text{ with } n = FDR.STEP = 0\text{-}1023 \quad (8.25)$$

8.1.7.2 Programmable Frequency Output for EXTCLK1

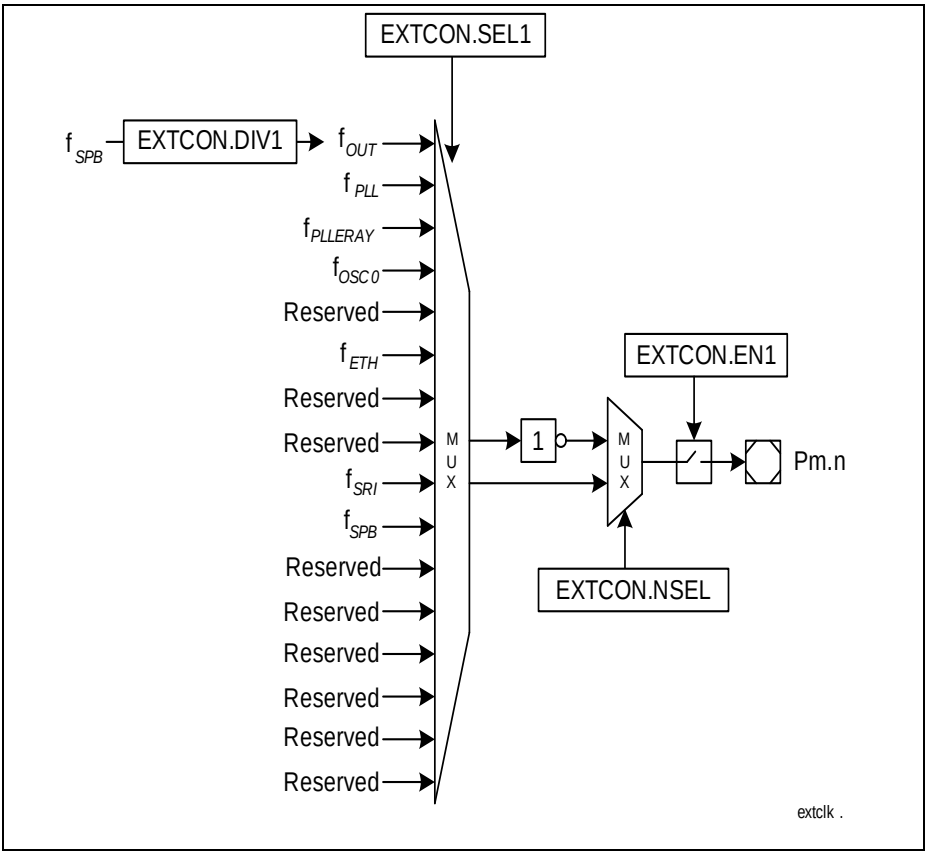


Figure 8-16 EXTCLK1 Generation

Clock f_{OUT} is generated via a counter, so the output frequency can be selected in small steps.

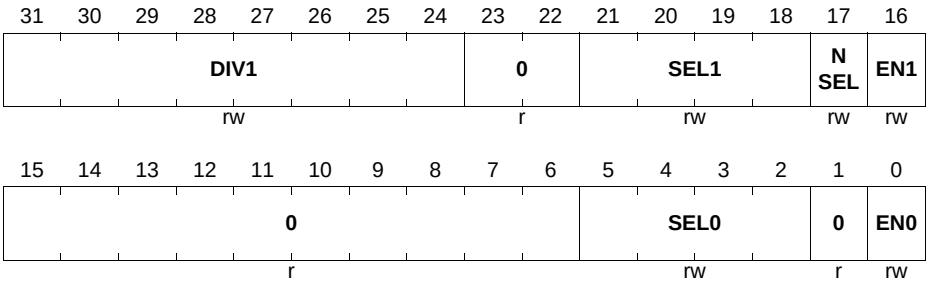
f_{OUT} always provides complete output periods.

Register EXTCON provides control over the output generation (frequency, activation).

8.1.7.3 Clock Output Control Register

These registers can be accessed by all CPUs in the system. Anyway it is suggested that only one CPU is used to control the clocks. As CPU0 is the active and available CPU after each reset this is the best choice.

EXTCON
External Clock Control Register ($03C_H$) Reset Value: $0000\ 0000_H$

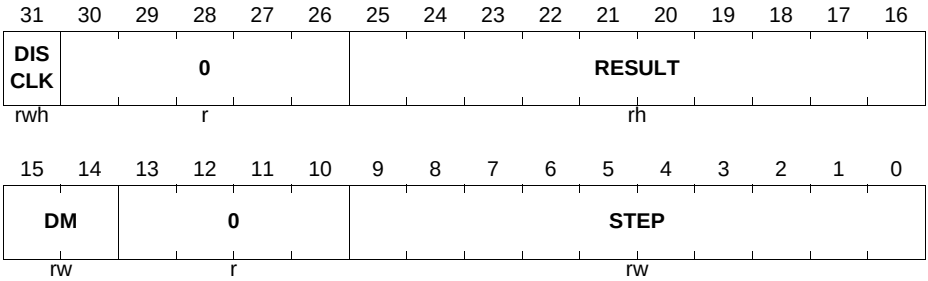


Field	Bits	Type	Description
EN0	0	rw	**External Clock Enable for EXTCLK0** 0_B No external clock is provided 1_B The configured external clock is provided

Field	Bits	Type	Description
SEL0	[5:2]	rw	**External Clock Select for EXTCLK0** This bit field defines the clock source that is selected as output for pin EXTCLK0. 0000_B f_{OUT} is selected for the external clock signal 0001_B f_{PLL} is selected for the external clock signal 0010_B $f_{PLLERAY}$ is selected for the external clock signal 0011_B f_{OSC0} is selected for the external clock signal 0100_B Reserved, do not use this combination 0101_B Reserved, do not use this combination 0110_B Reserved, do not use this combination 0111_B Reserved, do not use this combination 1000_B f_{SRI} is selected for the external clock signal 1001_B f_{SPB} is selected for the external clock signal 1010_B Reserved, do not use this combination 1011_B f_{STM} is selected for the external clock signal 1100_B Reserved, do not use this combination 1101_B Reserved, do not use this combination 1110_B Reserved, do not use this combination 1111_B f_{MT0} from the ERAY module is selected for the external clock signal
EN1	16	rw	**External Clock Enable for $\overline{\text{EXTCLK1}}$** 0_B No external clock is provided 1_B The configured external clock is provided
NSEL	17	rw	**Negation Selection** 0_B The external clock is inverted 1_B The external clock is not inverted

Field	Bits	Type	Description
SEL1	[21:18]	rw	**External Clock Select for EXTCLK1** This bit field defines the clock source that is selected as output for pin EXTCLK1. 0000_B f_{OUT} is selected for the external clock signal 0001_B f_{PLL} is selected for the external clock signal 0010_B $f_{PLLERAY}$ is selected for the external clock signal 0011_B f_{OSC0} is selected for the external clock signal 0100_B Reserved, do not use this combination 0101_B f_{ETH} is selected for the external clock signal 0110_B Reserved, do not use this combination 0111_B Reserved, do not use this combination 1000_B f_{SRI} is selected for the external clock signal 1001_B f_{SPB} is selected for the external clock signal 1010_B Reserved, do not use this combination 1011_B Reserved, do not use this combination 1100_B Reserved, do not use this combination 1101_B Reserved, do not use this combination 1110_B Reserved, do not use this combination 1111_B Reserved, do not use this combination
DIV1	[31:24]	rw	**External Clock Divider for EXTCLK1** This value defines the reload value of the divider that generates f_{OUT} out of f_{SPB} ($f_{OUT} = f_{SPB}/(DIV1+1)$). The divider itself is cleared each time bit EN1 is cleared.
0	1, [15:6], [23:22]	r	**Reserved** Read as 0; should be written with 0.

FDR
Fractional Divider Register **(038_H)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
STEP	[9:0]	rw	**Step Value** In Normal Divider Mode, STEP contains the reload value for RESULT. In Fractional Divider Mode, this bit field determines the 10-bit value that is added to RESULT with each input clock cycle.
DM	[15:14]	rw	**Divider Mode** This bit fields determines the functionality of the fractional divider block. 00_B Fractional divider is switched off; no output clock is generated. The Reset External Divider signal is 1. RESULT is not updated (default after System Reset). 01_B Normal Divider Mode selected. 10_B Fractional Divider Mode selected. 11_B Fractional divider is switched off; no output clock is generated. RESULT is not updated.
RESULT	[25:16]	rh	**Result Value** In Normal Divider Mode, RESULT acts as reload counter (addition +1). In Fractional Divider Mode, this bit field contains the result of the addition RESULT + STEP. If DM is written with 01_B or 10_B, RESULT is loaded with $3FF_H$.
DISCLK	31	rwh	**Disable Clock** 0_B Clock generation of f_{OUT} is enabled according to the setting of bit field DM. 1_B Fractional divider is stopped. No change except when writing bit field DM.
0	[13:10], [30:26]	r	**Reserved** Read as 0; should be written with 0.

8.1.8 Clock Generation Unit

The Clock Generation Unit (CGU) allows a very flexible clock generation for the TC21x/TC22x/TC23x. During user program execution the frequency can be programmed for an optimal ratio between performance and power consumption.

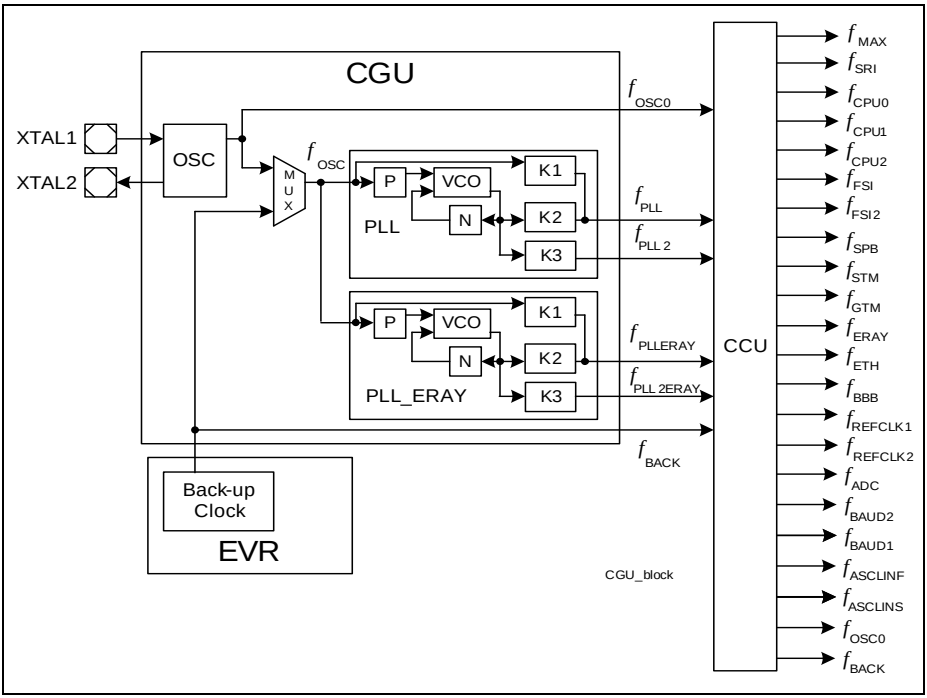


Figure 8-17 Clock Generation Unit Block Diagram

The CGU includes the clock source generation, the clock upscaling beside the clock distribution.

8.1.8.1 Example Sequence

The following sequence give an example for the clock ramp-up.

Goal for normal target setting is that system clock is based on PLL with external crystal.

- After power-on and system reset:
  - the system operates on the back-up clock
  - the oscillator (OSC) needs to be enabled via software or was enabled by firmware via BMI settings
- Fast clocks (SRI-Bus and CPUs) run with approximately 100 MHz

- Peripherals and SPB-Bus run with approximately 50 MHz
- Step 1: If f_{OSC0} is to be used but is not running, enable the oscillator to wait until it is providing a stable clock
- Step 2: Initialize the PLL to target f_{VCO} and f_{PLL} frequency
  - Select PLL input clock via CCUCON1.INSEL
  - Disconnect input clock from VCO; set PLLCON0.SETFINDIS
  - Select P, K2 / K3, and N divider for final target VCO and PLL frequency
  - 1) Select P as small as possible
  - 2) Select N (VCO) as high as possible
  - Connect input clock to VCO; set PLLCON0.CLRFINDIS
  - Configure resulting K2
- Step 3: Wait for PLL lock to be set
- Step 4: Configure CCUCON0, CCUCON1 and CCUCON5 to first target setting
  - Hint: both registers CCUCON0 and CCUCON1 can be reprogrammed without changing the clocking. Only setting the UP bit in one of the two registers transfers the values for both registers and lead to changes in the clock system.
  - Register CCUCON2 has its own UP bit
  - Bit fields CCUCON1.ERAY and CCUCON1.ETH are excluded from the update via UP in CCUCON0 or CCUCON1 and are always updated when a new value is programmed
- Step 5: Switch CCU input clock f_{SOURCE} to PLL via CCUCON0.CLKSEL
- Step 6: After setting CCU f_{SOURCE} to f_{PLL}, frequency has to be increased step by step to target frequency

Important hints for steps 4 and 5;

- Select CCUCON1 and CCUCON0 register settings in a way that the current consumption of the 1.3 V domain does not change more than the desired target value of the application, e.g. 50 mA.

- After every frequency programming step a wait time is recommended until supply ripple caused by supply current transient is faded away.

Note: The wait time between frequency steps depends on the supply and block concept.

- Continue these CCUCON1 and CCUCON0 update steps until the target system frequency is reached.

8.1.9 CCU Register Address

Table 8-5 Registers Address Spaces - CCU Registers

Module	Base Address	End Address	Note
SCU	$F003\ 6000_H$	$F003\ 63FF_H$	-

8.1.10 CCU Kernel Registers

This section describes the CCU kernel registers of the SCU module. Most of CCU kernel register names described in this section will be referenced in other parts of the TC21x/TC22x/TC23x User's Manual by the module name prefix "SCU_".

CCU Kernel Register Overview

Table 8-6 Register Overview of CCU Registers (Offset from SCU Base)

Short Name	Long Name	Offset Addr. [1]	Access Mode		Reset	Description See
			Read	Write		
OSCCON	OSC Control Register	010_H	U, SV	SV, SE, P	System Reset	Page 8-5
PLLSTAT	PLL Status Register	014_H	U, SV	BE	System Reset	Page 8-20
PLLCON0	PLL Configuration 0 Register	018_H	U, SV	SV, SE, P	System Reset	Page 8-21
PLLCON1	PLL Configuration 1 Register	$01C_H$	U, SV	SV, SE, P	System Reset	Page 8-23
PLLCON2	PLL Configuration 2 Register	020_H	U, SV	SV, SE, P	System Reset	Page 8-24
PLLERAYSTAT	PLL_ERAY Status Register	024_H	U, SV	BE	System Reset	Page 8-33
PLLERAYCON0	PLL_ERAY Configuration 0 Register	028_H	U, SV	SV, SE, P	System Reset	Page 8-35
PLLERAYCON1	PLL_ERAY Configuration 1 Register	$02C_H$	U, SV	SV, SE, P	System Reset	Page 8-36

Table 8-6 Register Overview of CCU Registers (Offset from SCU Base)

Short Name	Long Name	Offset Addr. 1)	Access Mode		Reset	Description See
			Read	Write		
CCUCON0	CCU Control Register 0	030_H	U, SV	SV, SE, P	System Reset	Page 8-43
CCUCON1	CCU Control Register 1	034_H	U, SV	SV, SE, P	System Reset	Page 8-48
FDR	Fractional Divider Register	038_H	U, SV	SV, SE, P	System Reset	Page 8-77
EXTCON	External Clock Control Register	$03C_H$	U, SV	SV, SE, P	System Reset	Page 8-75
CCUCON2	CCU Control Register 2	040_H	U, SV	SV, SE, P	System Reset	Page 8-57
CCUCON3	CCU Control Register 3	044_H	U, SV	SV, SE, P	System Reset	Page 8-66
CCUCON4	CCU Control Register 4	048_H	U, SV	SV, SE, P	System Reset	Page 8-68
CCUCON5	CCU Control Register 5	$04C_H$	U, SV	SV, SE, P	System Reset	Page 8-55
CCUCON6	CCU Control Register 6	080_H	U, SV	SV, SE, P	System Reset	Page 8-59
–	Reserved	244_H - $3F4_H$	BE	BE	–	–

1) The absolute register address is calculated as follows:
Module Base Address + Offset Address (shown in this column)

8.2 Reset Control Unit (RCU)

The Reset Control Unit (RCU) of the TC21x/TC22x/TC23x contains the following functional sub-blocks:

- Basic Reset Operation (see **Section 8.2.1**)
- External Reset sources and indications (see **Section 8.2.2**)
- Software Boot Support (see **Section 8.2.3**)
- NMI Trap Generation (see **Section 8.2.4**)
- RCU register overview table (see **Table 8-17**)

8.2.1 Reset Operation

This section describes the conditions under which the TC21x/TC22x/TC23x will be reset and the reset operation configuration and control.

8.2.1.1 Overview

The following reset request triggers are available:

- Supply monitor (SWD) triggers a power-on reset (cold reset)
- 1.3V EVR monitor triggers a power-on reset (cold reset)
- Standby EVR (STBYR) monitor triggers a power-on reset (cold reset)
- External active low hardware “power-on” reset request trigger; $\overline{\text{PORST}}$ (can be either a warm reset or to extend a cold reset)
- External System Request reset trigger pins; $\overline{\text{ESR0}}$ and ESR1 (warm reset)
- Safety Management Unit (SMU) alarm reset request trigger, (warm reset)
- Software reset (SW), (warm reset)
- System Timer (STMx) trigger (warm reset)
- Resets via the JTAG interface
- JTAG resets (special reset)
- Software triggered module reset
-
-
-

Note: The JTAG resets are described in the OCDS chapter.

8.2.1.2 Reset Types

The following list describes the different reset types.

- Power-on Reset :
  This reset results in initialization of the complete system into a defined state. A Power-on Reset also generates a Debug Reset and a System Reset (and therefore also an Application Reset).
- System Reset :
  This reset leads to a initialization into a defined state of the complete system but without a reset of the power subsystem, debug subsystem or reset configuration registers.
  A System Reset also generates an Application Reset.
- Debug Reset :
  This reset leads to a initialization into a defined state of the complete debug system.
- Application Reset:
  This reset leads to a initialization into a defined state of the complete application system with the following parts: all peripherals, the CPUs and parts of the SCU.

- Module Resets:
  Module resets result in individual modules being initialized into a defined state without any impact on the rest of the system.

8.2.1.3 Reset Sources Overview

The connection of the reset sources and the activated reset signals/domains are shown in **Table 8-7**.

Table 8-7 Effect of Reset Triggers

Reset Request Trigger	Application Reset	Debug Reset	System Reset
$\overline{\text{PORST}}$	Activated	Activated	Activated
$\overline{\text{STBYR}}$	Activated	Activated	Activated
$\overline{\text{SWD}}$	Activated	Activated	Activated
$\overline{\text{EVR13}}$	Activated	Activated	Activated
$\overline{\text{ESR0}}$	Configurable	Not Activated	Configurable
$\overline{\text{ESR1}}$	Configurable	Not Activated	Configurable
SMU	Configurable	Not Activated	Configurable
STM0	Configurable	Not Activated	Configurable
SW	Configurable	Not Activated	Configurable
Cerberus RSTCL0[1)]	Activated	Not Activated	Activated
Cerberus RSTCL1[2)]	Not Activated	Activated	Not Activated
Cerberus RSTCL3[3)]	Activated	Not Activated	Not Activated
Module Resets[4)]	Not Activated	Not Activated	Not Activated

1) Cerberus resets may be triggered by CBS_OCNTRL or CBS_OJCONF
2) Cerberus resets may be triggered by CBS_OCNTRL or CBS_OJCONF
3) Cerberus resets may be triggered by CBS_OCNTRL or CBS_OJCONF
4) Module Resets (via MOD_KRSTx.RST register bits in some modules) have no effect on chip-level reset system. The scope of a module reset is limited to the module itself.

8.2.1.4 Warm and Cold Resets

A cold “power-on” reset is a reset which is triggered for the first time during a system power-up or in response to a temporary power failure. During the power-up the EVR primary undervoltage monitors will trigger a complete reset of the system, placing the system into a known state. The $\overline{\text{PORST}}$ pin can be used to extend this reset phase and control the timing of its release. The pins and internal states are placed immediately into their reset state when the trigger is asserted.

A warm reset is a reset which is triggered while the system is already operational and the supplies remain stable. It is used to return the system to the same known state. On a warm reset request, any reset of pin states will take place immediately, but the internal circuitry will only be reset after the system has been brought into a state where memory contents and debug trace data will not be corrupted and the current consumption has been ramped down. Note that $\overline{\text{PORST}}$ may be asserted as a warm reset.

8.2.1.5 EVR Resets and PORST

The $\overline{\text{PORST}}$ pin is a bidirectional reset in/output intended for external triggering of power-related resets. If this pin is left open then the default behavior shall be that the device remains in a reset state. If the $\overline{\text{PORST}}$ pin remains asserted after a power event then the reset will be extended until it is deasserted. This does not replace the ESR pins functional reset.

8.2.1.6 Module Reset Behavior

Table 8-8 lists how the various functions of the TC21x/TC22x/TC23x are affected by each reset type. An “X” means that this block has at least some register/bits that are affected by the corresponding reset.

Table 8-8 Effect of Reset on Device Functions

Module / Function	Application Reset	Debug Reset	System Reset	Warm “Power-on” Reset	Cold Power-on Reset
All TriCore CPU Core	X	X	X	X	X
Peripherals (except SCU)	X	X	X	X	X
SCU	X	Not affected	X	X	X
Flash Memory	Not affected	Not affected, reliable	X	X	X
JTAG Interface	Not affected	Not affected	Not affected	X	X
OCDS	Not affected	X	Not affected	X	X
MCDS	Not affected	X	Not affected	X	X
Backup Clock	Not affected	Not affected	Not affected	Not affected	Not affected

Table 8-8 Effect of Reset on Device Functions (cont'd)

Module / Function	Application Reset	Debug Reset	System Reset	Warm "Power-on" Reset	Cold Power-on Reset
XTAL Oscillator, PLLs	Not affected	Not affected	X	X	X
Port Pins	X	Not affected	X	X	X
Pins ESRx	Not affected	Not affected	X	X	X
EVR	Not affected	Not affected	Not affected	Not affected	X
On-chip Static RAMs[1)]					
CAN	Initialized	Initialized	Initialized	Initialized	Uninitialized
All DSPRs[2)]	Not affected, reliable	Not affected, reliable	Not affected, reliable	Not affected, reliable	Uninitialized
All PSPRs[3)]	Not affected, reliable	Not affected, reliable	Not affected, reliable	Not affected, reliable	Uninitialized
All PCACHEs and DCACHES[4)]	Cache invalidated	Cache invalidated	Cache invalidated	Cache invalidated	Uninitialized
LMU[5)]	Not affected, reliable	Not affected, reliable	Not affected, reliable	Not affected, reliable	Uninitialized
Other	Not affected, reliable	Not affected, reliable	Not affected, reliable	Not affected, reliable	Uninitialized

1) Reliable here means that also the redundancy is not affected by the reset.

2) DSPR is partially used as a scratchpad by the startup firmware. Previous data stored in the upper 32kB will be overwritten on start-up. Auto-initialization on reset can be optionally configured by PROCOND.RAMIN and PROCOND.RAMINSEL. Auto-initialization is disabled for CPU0 DSPR if the standby RAM is enabled.

3) Depending upon PROCOND.RAMIN and RAMINSEL setting, these memories may be automatically erased on cold or warm resets.

4) Tag memories are invalidated by hardware at reset. Depending upon PROCOND.RAMIN setting, cache content may also be automatically erased at reset (See MTU chapter for details).

5) Depending upon PROCOND.RAMIN and RAMINSEL setting, these memories may be automatically erased on cold or warm resets.

8.2.1.7 General Reset Operation

A reset is generated if an enabled reset request trigger is asserted. Most reset triggers can be configured to initiate different types of reset. No action (disabled) is a valid

configuration which is selected by setting the corresponding bit field in the Reset Configuration Register to 00_B.

A Debug Reset can only be requested by dedicated reset request triggers and can not be selected via a Reset Configuration Register . For more information see also register RSTCON.

8.2.1.8 Reset Generation

The figure below shows the RCU controlling the reset generation for the complete device, with the exception of the JTAG reset domain.

Note: The JTAG reset domain is controlled by the $\overline{TRST}$ pin.

The RCU detects the different reset requests, schedules a shutdown of the system, and then generates the appropriate internal reset pulse(s).

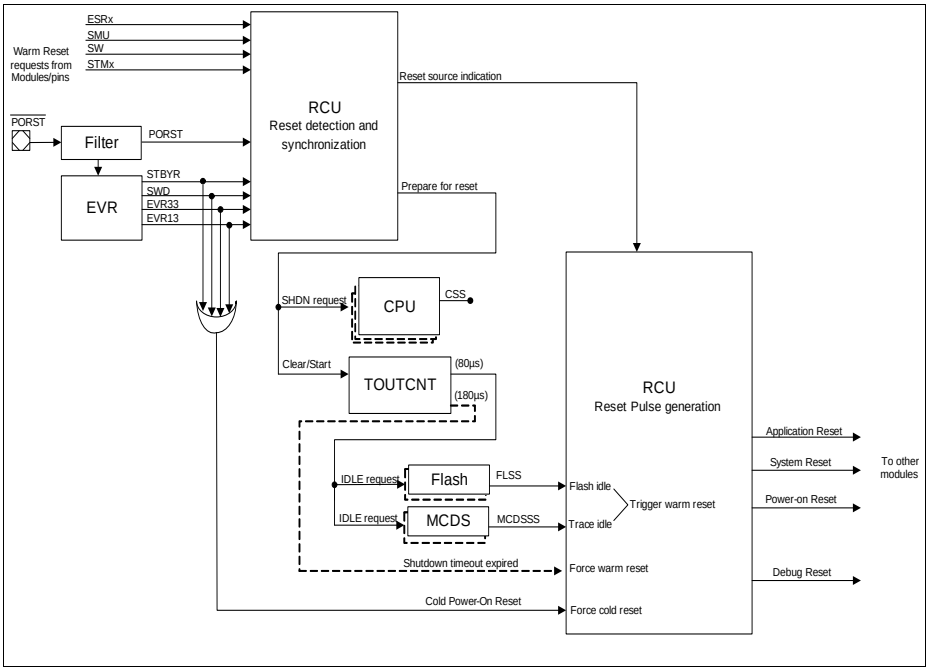


Figure 8-18 Reset Overview

8.2.1.9 Shutdown and Reset Delay Timeout Counter (TOUTCNT)

The system is automatically brought to a stable state before the internal reset(s) are applied so that the RAM content may be reliably reused after a warm reset

On detection of a warm Application Reset, System Reset or Power-On Reset trigger, a Shutdown Trap request is sent by the RCU to the CPU(s). This request causes the CPU(s) which are not already in a halt or idle state to unconditionally execute a ROM shutdown routine which executes a controlled rampdown of the device power consumption (to prevent current jumps which might trigger EVR reset). At the end of this routine each CPU executes a WAIT instruction and stalls, to ensure that all ongoing write transactions have been completed. Interrupts and NMI to the CPUs during the shutdown routine will not reawaken CPUs.

The internal reset starts when all relevant system modules are Idle. If timeout counter TOUTCNT exceeds a timeout period (180 microseconds) before the system become idle then the reset is started regardless. This timeout ensures that a reset would still occur in the case of an internal deadlock.

The RSTCON2.CSS bits indicate whether each CPU successfully flushed its write buffers and reached an idle state before the previous reset. These bits can therefore be used to determine whether RAM content integrity can be trusted after the previous reset cycle.

8.2.1.10 Reset Triggers

There are two types of reset triggers for the reset control logic:

- Triggers that lead to a specific reset
- Triggers that lead to a configurable reset

Specific Reset Triggers

Assertion of these triggers leads to a predefined reset type. These triggers can not be enabled / disabled. All specific reset triggers are listed in **Table 8-9**.

Table 8-9 Specific Reset Triggers

Reset Trigger	Reset Type Request
Cerberus 0 (CB0)	SystemReset
Cerberus 1 (CB1)	DebugReset
Cerberus 3 (CB3)	ApplicationReset
STBYR	Power-on Reset
SWD	Power-on Reset
EVR13	Power-on Reset

Configurable Reset Triggers

Assertion of these triggers leads to a configurable reset type. These triggers can be enabled / disabled. The result of the reset triggers is defined by the corresponding bit field in register RSTCON.

Prevention of Double SMU Resets

After boot firmware initialization, the default behaviour of the SMU/RCU is that a Watchdog timeout would result in an NMI followed by an Application Reset. If an SMU-induced Application Reset occurs twice, a severe system malfunction is assumed and the TC21x/TC22x/TC23x is held in permanent Application Reset until a Power-On or System Reset occurs. This prevents the device from being periodically reset if the application fails to service the watchdog correctly and the watchdog repeatedly times out.

An internal flag is set when the first SMU reset is requested. If a second reset is also requested by the SMU when the internal flag is already set, the double SMU reset event has occurred and a permanent reset request is automatically generated. This internal flag is only reset by a System Reset or when bit WDTSCON1.CLRIRF is set and bit WDTSCON0.ENDINIT has also been set. A correct service of the WDT does not automatically clear this flag.

If this behaviour is undesirable, then the CLRIRF bit should be written by the application during initialization (before the watchdog times out), to clear the flag and prevent any subsequent occurrence of a permanent reset.

Note: If for any reason random code is executed bit field RSTCON.SMU can be updated unintentionally. This can result in an SMU alarm not leading to a reset. To avoid this after the SSW is finished this bit field should be checked and the Safety WDT ENDINIT protection enabled.

8.2.1.11 Debug Reset Specific Behavior

For safety reasons it is required by the debugger that if the OCDS system is disabled a DebugReset is also asserted every time an Application Reset is asserted.

8.2.1.12 Module Resets

Many modules within TC21x/TC22x/TC23x can be reset individually. The module reset has no effect outside the module itself. A module reset can only be triggered by writing ‘1’ into both of the module reset registers MOD_KRST1.RST and register MOD_KRST0.RST.

The Module Reset register bits may only be written by those masters which have been explicitly configured to have module access (See module ACCEN register). In addition, it is only possible to write to these reset bits during the short time window after a correct

ENDINIT password unlock sequence (See Watchdog chapter for details). This prevents accidental module resets by rogue software.

The table below shows the modules which have Module Reset capability

Table 8-10 Module Resets

Module	Module Reset capability
CAN	Module reset implemented in CAN
ASCLIN	Module reset implemented in ASCLIN
Flexray	Module reset implemented in ERAY
QSPI	Module reset implemented in QSPI
ETHERMAC	Module reset implemented in ETHERMAC
SENT	Module reset implemented in SENT
VADC	Module reset implemented in VADC
CCU6	Module reset implemented in CCU6
GPT12	Module reset implemented in GPT12
SMU	Module reset implemented in SMU
STM	Module reset implemented in STM
DMA	Per channel reset implemented in DMA
SCU, RCU, PMC, CCU	No module reset capability
PMU, Flash	No module reset capability
LMU	No module reset capability
PORTS	No module reset capability
IR	No module reset capability
IOM	Module reset implemented in IOM

8.2.1.13 Reset Controller Registers

Status Registers

After a chip level reset has been executed, the Reset Status registers provide information on the trigger of the last reset(s). Warm reset status bits are updated upon each reset cycle. A reset cycle is finished when all resets are de-asserted. Within a reset cycle the status flags are used as sticky flags. Module level resets have no effect on the Reset Status register. Bits which may indicate a cold reset trigger (SWD, PORST and EVR13) provide useful information to the application and are not cleared automatically. The application may clear these bits by writing to bit RSTCON2.CLRC.

RSTSTAT
Reset Status Register

(050_H) **Reset Value: $1381\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			STBYR	0		SWD	EVR33	EVR13	0		CB3	CB1	CB0	0	PORST
r			rh	r		rh	rh	rh	r		rh	rh	rh	r	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								STM2	STM1	STM0	SW	SMU	0	ESR1	ESR0
r								rh	rh	rh	rh	rh	r	rh	rh

Field	Bits	Type	Description
ESR0	0	rh	**Reset Request Trigger Reset Status for ESR0** 0_B The last reset was not requested by this reset trigger 1_B The last reset was requested by this reset trigger
ESR1	1	rh	**Reset Request Trigger Reset Status for ESR1** 0_B The last reset was not requested by this reset trigger 1_B The last reset was requested by this reset trigger

Field	Bits	Type	Description
SMU	3	rh	**Reset Request Trigger Reset Status for SMU** (See SMU section for SMU trigger sources, including Watchdog Timers) 0_B The last reset was not requested by this reset trigger 1_B The last reset was requested by this reset trigger
SW	4	rh	**Reset Request Trigger Reset Status for SW** 0_B The last reset was not requested by this reset trigger 1_B The last reset was requested by this reset trigger
STM0	5	rh	**Reset Request Trigger Reset Status for STM0 Compare Match** 0_B The last reset was not requested by this reset trigger 1_B The last reset was requested by this reset trigger
STM1	6	rh	**Reset Request Trigger Reset Status for STM1 Compare Match (If Product has STM1)** 0_B The last reset was not requested by this reset trigger 1_B The last reset was requested by this reset trigger
STM2	7	rh	**Reset Request Trigger Reset Status for STM2 Compare Match (If Product has STM2)** 0_B The last reset was not requested by this reset trigger 1_B The last reset was requested by this reset trigger
PORST	16	rh	**Reset Request Trigger Reset Status for PORST** 0_B This reset trigger has not occured since the last clear (by RSTCON2.CLRC) 1_B This reset trigger has occured since the last clear (by RSTCON2.CLRC) This bit is also set if the bits CB0, CB1, and CB3 are set in parallel.
0	17	r	**Reserved** Read as 0; should be written with 0.

Field	Bits	Type	Description
CB0	18	rh	**Reset Request Trigger Reset Status for Cerberus System Reset** 0_B The last reset was not requested by this reset trigger 1_B The last reset was requested by this reset trigger
CB1	19	rh	**Reset Request Trigger Reset Status for Cerberus Debug Reset** 0_B The last reset was not requested by this reset trigger 1_B The last reset was requested by this reset trigger
CB3	20	rh	**Reset Request Trigger Reset Status for Cerberus Application Reset** 0_B The last reset was not requested by this reset trigger 1_B The last reset was requested by this reset trigger
EVR13	23	rh	**Reset Request Trigger Reset Status for EVR13** 0_B This reset trigger has not occured since the last clear (by RSTCON2.CLRC) 1_B This reset trigger has occured since the last clear (by RSTCON2.CLRC)
EVR33	24	rh	**Reserved in this product**
SWD	25	rh	**Reset Request Trigger Reset Status for Supply Watchdog (SWD)** The Supply Watchdog trigger is described in Power Management Controller “Supply Monitoring” chapter 0_B This reset trigger has not occured since the last clear (by RSTCON2.CLRC) 1_B This reset trigger has occured since the last clear (by RSTCON2.CLRC)
STBYR	28	rh	**Reset Request Trigger Reset Status for Standby Regulator Watchdog (STBYR)** 0_B This reset trigger has not occured since the last clear (by RSTCON2.CLRC) 1_B This reset trigger has occured since the last clear (by RSTCON2.CLRC)

Field	Bits	Type	Description
0	[15:8], [31:29] ,[27:26],21,22 ,2	r	**Reserved** Read as 0; should be written with 0.

Reset Configuration Registers

RSTCON
Reset Configuration Register

(058$_H$) **Reset Value: 0000 0282$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STM2		**STM1**		**STM0**		**SW**		**SMU**		**0**		**ESR1**		**ESR0**	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
ESR0	[1:0]	rw	**ESR0 Reset Request Trigger Reset Configuration** This bit field defines which reset is generated by a reset request trigger from ESR0 reset. 00_B No reset is generated for a trigger of ESR0 01_B A System Reset is generated for a trigger of ESR0 reset 10_B An Application Reset is generated for a trigger of ESR0 reset 11_B Reserved, do not use this combination

Field	Bits	Type	Description
ESR1	[3:2]	rw	**ESR1 Reset Request Trigger Reset Configuration** This bit field defines which reset is generated by a reset request trigger from ESR1 reset. 00_B No reset is generated for a trigger of ESR1 01_B A System Reset is generated for a trigger of ESR1 reset 10_B An Application Reset is generated for a trigger of ESR1 reset 11_B Reserved, do not use this combination
0	[5:4]	rw	**Reserved** Should be written with 0.
SMU	[7:6]	rw	**SMU Reset Request Trigger Reset Configuration** This bit field defines which reset is generated by a reset request trigger from SMU reset. 00_B No reset is generated for a trigger of SMU 01_B A System Reset is generated for a trigger of SMU reset 10_B An Application Reset is generated for a trigger of SMU reset 11_B Reserved, do not use this combination
SW	[9:8]	rw	**SW Reset Request Trigger Reset Configuration** This bit field defines which reset is generated by a reset request trigger from software reset. 00_B No reset is generated for a trigger of software reset 01_B A System Reset is generated for a trigger of Software reset 10_B An Application Reset is generated for a trigger of Software reset 11_B Reserved, do not use this combination
STM0	[11:10]	rw	**STM0 Reset Request Trigger Reset Configuration** This bit field defines which reset is generated by a reset request trigger from STM0 compare match reset. 00_B No reset is generated for an STM0 trigger 01_B A System Reset is generated for a trigger of STM0 reset 10_B An Application Reset is generated for a trigger of STM0 reset 11_B Reserved, do not use this combination

Field	Bits	Type	Description
STM1	[13:12]	rw	**STM1 Reset Request Trigger Reset Configuration (If Product has STM1)** This bit field defines which reset is generated by a reset request trigger from STM1 compare match reset. 00_B No reset is generated for a trigger of STM1 01_B A System Reset is generated for a trigger of STM1 reset 10_B An Application Reset is generated for a trigger of STM1 reset 11_B Reserved, do not use this combination
STM2	[15:14]	rw	**STM2 Reset Request Trigger Reset Configuration (If Product has STM2)** This bit field defines which reset is generated by a reset request trigger from STM2 compare match reset. 00_B No reset is generated for a trigger of STM2 01_B A System Reset is generated for a trigger of STM2 reset 10_B An Application Reset is generated for a trigger of STM2 reset 11_B Reserved, do not use this combination
0	[31:16]	rw	**Reserved** Should be written with 0.

ARSTDIS
Application Reset Disable Register

(05C$_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								0					STM 2DIS	STM 1DIS	STM 0DIS
r								rw					rw	rw	rw

Field	Bits	Type	Description
STM0DIS	0	rw	**STM0 Disable Reset** This bit field defines if an Application Reset leads to an reset for the STM0. 0_B An Application Reset resets the STM0 1_B An Application Reset has no effect for the STM0
STM1DIS	1	rw	**STM1 Disable Reset (If Product has STM1)** This bit field defines if an Application Reset leads to an reset for the STM1. 0_B An Application Reset resets the STM1 1_B An Application Reset has no effect for the STM1
STM2DIS	2	rw	**STM2 Disable Reset (If Product has STM2)** This bit field defines if an Application Reset leads to an reset for the STM2. 0_B An Application Reset resets the STM2 1_B An Application Reset has no effect for the STM2
0	[31:8]	r	**Reserved** Read as 0; should be written with 0.
0	[7:3]	rw	**Reserved** Read as 0; should be written with 0.

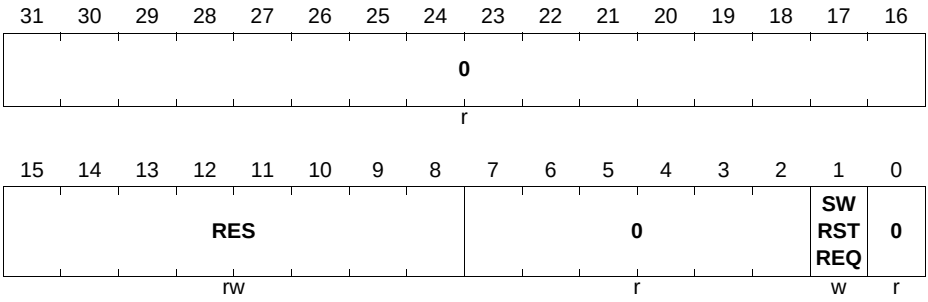
SW Reset Configuration Register

This register controls the SW Reset operation.

As for other kernel registers, write access to the SWRSTCON register is configurable via SCU_ACCEN0 and additionally is temporally restricted by Safe ENDINIT. These restrictions can be used to provide protection against accidental reset by unauthorised CPUs or system bus masters.

SWRSTCON
Software Reset Configuration Register

(060_H) **Reset Value: 0000 XX00$_H$**

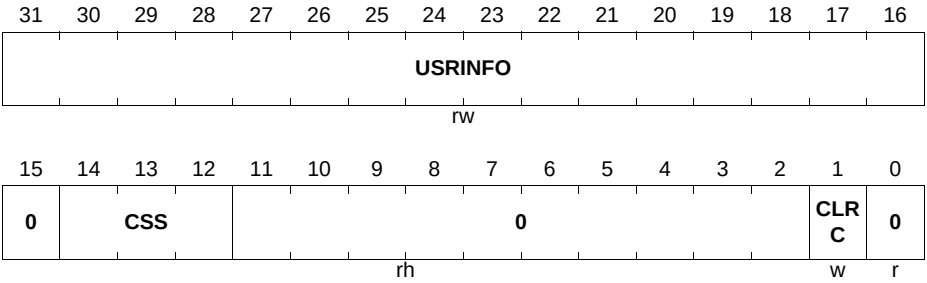


Field	Bits	Type	Description
0	0	rw	**Reserved** Read as 0; should be written with 0.
SWRSTREQ	1	w	**Software Reset Request** 0_B No SW Reset is requested 1_B A SW Reset request trigger is generated This bit is automatically cleared and read always as zero.
RES	[15:8]	rw	**Reserved** Should be written with original content.
0	[7:2], [31:16]	r	**Reserved** Read as 0; should be written with 0.

Additional Reset Control Register

RSTCON2
Additional Reset Control Register

(064_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
USRINFO	[31:16]	rw	**User Information** User data register (Cleared only on Power-on reset).
Reserved	[11:2], 15	r	**Reserved** Internal use only. Bits may read as 0 or 1. Writes have no effect.
CSS2	14	r	**Reserved in this product** Bit reads as ‘1’ after warm reset
CSS1	13	r	**Reserved in this product** Bit reads as ‘1’ after warm reset
CSS0	12	rh	**CPU0 Safe State Reached** The state of CPU0 before the last warm reset If any bit is zero after an Application Reset (or higher) then it is possible that SRAM content could have been corrupted by the reset 0_B CPU0 safe state not achieved prior to last reset 1_B CPU0 in safe state at last reset

Field	Bits	Type	Description
CLRC	1	w	**Clear Cold Reset Status** This bit simultaneously clears the sticky status bits which may indicate any previous cold reset (i.e. RSTSTAT.STBYR, RSTSTAT.SWD, RSTSTAT.EVR33, RSTSTAT.EVR13 and RSTSTAT.PORST). 0_B No effect 1_B Clear cold reset RSTSTAT status bits
0	0	r	**Reserved** Bits read as 0. Should only be written as 0.

8.2.2 External Reset Sources and Indications

ESR interface pins are provided to give an external indication of internal resets, and to provide a mechanism for external sources to trigger internal resets:

ESR pin functions:

- Reset request trigger
- Wakeup trigger
- Reset indication output
- Trap request trigger

8.2.2.1 External Service Requests ($\overline{\text{ESR}}$x)

The ESR pins can be used in various ways:

- ESR inputs can trigger a reset
- ESR outputs can provide a reset indication
- ESR inputs can trigger a trap (NMI)
- ESR pins can be used as general data I/O pins
- ESR inputs can trigger a wake-up from standby mode

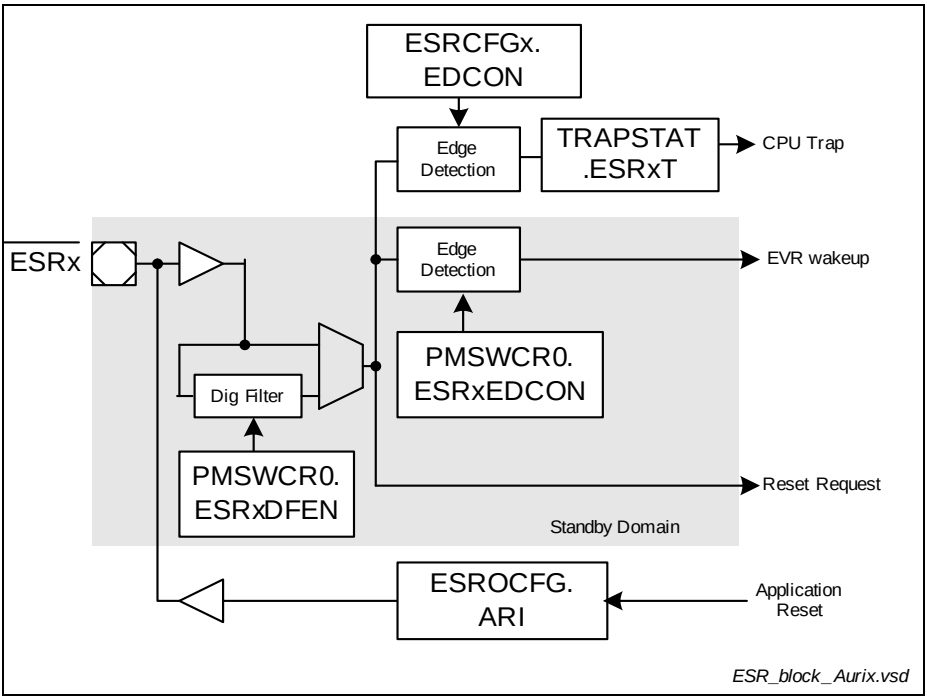


Figure 8-19 ESR Operation

ESRx as Reset Request Trigger

An $\overline{\text{ESR0}}$/$\overline{\text{ESR1}}$ pin reset request trigger can lead to a Systemor Application Reset. The type of the reset is configured via RSTCON.ESRx.

The input signals $\overline{\text{ESR0}}$/$\overline{\text{ESR1}}$ can be filtered. The filter can be disabled by register bit PMSWCR0.ESRxDFEN.

If the digital filter is enabled then pulses less than 20ns cannot trigger a reset and pulses longer than 100ns will always result in a trigger..

The behavior of $\overline{\text{ESR0}}$/$\overline{\text{ESR1}}$ pins can be configured by programming registers ESRCFG0/1. The pad control functionality can be configured independently for each pin. The configuration comprises:

- Selection of driver type (open-drain or push-pull)
- Enable of the output driver (input and/or output capability)
- Enable of internal pull-up or pull-down resistance

ESRx as Reset Output

The external pins $\overline{\text{ESR0}}$/$\overline{\text{ESR1}}$ can provide a reset output indication (open drain) for Application Resets .

Register ESROCFG.ARI determines the output of the ESR pin(s) when one or more are configured as functional reset outputs. ARI is set automatically when any Application Reset starts and cleared by Boot or Application software (Note that an Application Reset also occurs on a System Reset or Power-On Reset). While the ARI bit is set, the configured ESR pin(s) drive active low. A subsequent write to ESROCFG.ARC clears the ARI bit, which deasserts the ESR output(s). The exact duration of the ESR pulse is determined by the value of the Flash Config Sector setting “ESR0CNT” which is copied by Boot Code into FLASH0_PROCOND.ESR0CNT on a cold power-on reset. The effect of this value is shown in **Table 8-11**. When the same ESR pin is configured as a PORT output, its state can instead be controlled directly by application software so that the application has the possibility to reset external devices.

Do not configure ESR pin to be used as both an output (reset indication or general purpose port) and a reset trigger input simultaneously.

Table 8-11 ESRx Reset Indication Options

ESR0CNT value	ESR Reset Indication Behaviour
0000 0000h	ESR0 deasserted as soon as possible after start of Boot Code execution
> 0000 0000h and < FFFh	ESR0 deasserted after programmed delay (by Boot Code). Boot Code may be extended and Application start may be delayed if programmed delay exceeds the normal Boot Code execution time. Programmed delay is ESR0CNT * 10 microseconds
FFFh	ESR0 not deasserted by Boot Code. Application code must deassert ESROCFG.ARI bit or PORT output to deassert ESRx pin(s).

An external device may extend an existing reset condition by holding the ESR pin active.

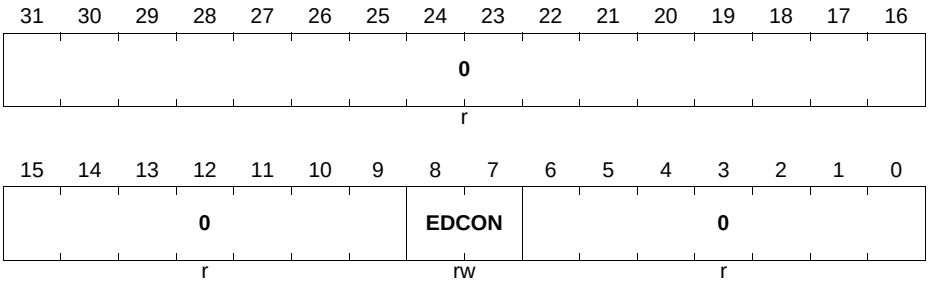
ESR Registers

ESRCFG0
ESR0 Input Configuration Register

(070_H) **Reset Value: $0000\ 0100_H$**

ESRCFG1
ESR1 Input Configuration Register

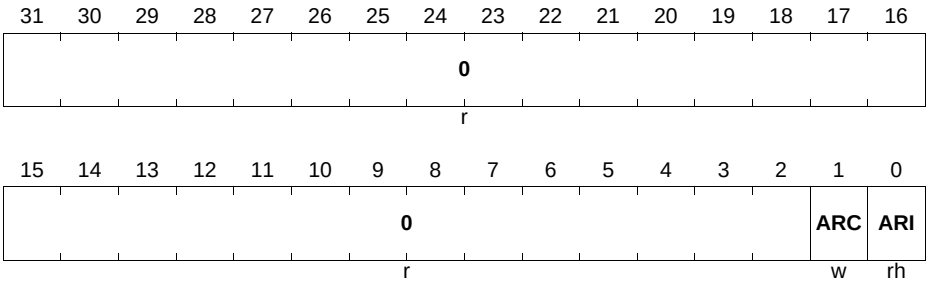
(074_H) **Reset Value: $0000\ 0100_H$**



Field	Bits	Type	Description
0	[6:0]	r	**Reserved** Read as 0; should be written with 0.
EDCON	[8:7]	rw	**Edge Detection Control** This bit field defines the edges that lead to an ESRx trigger of the synchronous path. 00_B No trigger is generated 01_B A trigger is generated upon a rising edge 10_B A trigger is generated upon a falling edge 11_B A trigger is generated upon a rising OR falling edge
0	[31:9]	r	**Reserved** Read as 0; should be written with 0.

ESROCFG
ESR Output Configuration Register

(078_H) **Reset Value: 0000 000X$_H$**

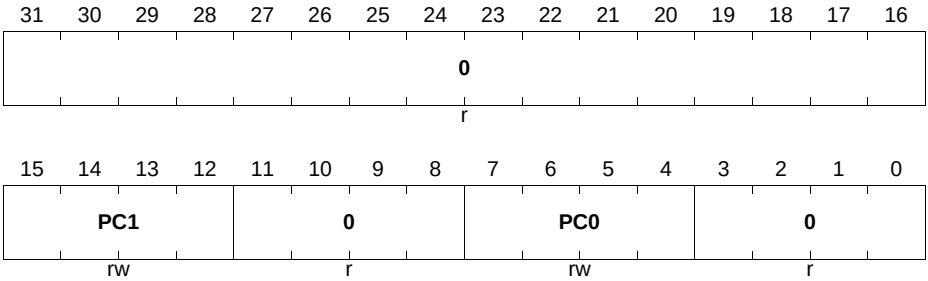


Field	Bits	Type	Description
ARI	0	rh	**Application Reset Indicator** This bit is set when an Application Reset request trigger occurs and cleared by writing to ARC. 0_B No application reset trigger detected (since last clear) 1_B Application reset trigger detected (since last clear) When the ARI bit is set and an ESR pin is configured as a reset output, the corresponding ESR input will not re-trigger a reset. This prevents feedback of the reset indication causing a new reset request. Extension of the reset by an external ESR source is handled by SSW. *Note: Observed reset value after boot will depend upon ARI mode, see **Table 8-11***
ARC	1	w	**Application Reset Indicator Clear** 0_B No effect 1_B Clear Application Reset Indicator (ARI) Read as 0
0	[31:2]	r	**Reserved** Read as 0; should be written with 0.

The input/output control registers select the digital output and input driver functionality and characteristics of the pin. Direction (input or output), pull-up or pull-down devices for inputs, and push-pull or open-drain functionality for outputs can be selected by the corresponding bit fields PCx (x = 0-1).

IOCR

Input/Output Control Register ($0A0_H$) Reset Value: 0000 $20E0_H$



Field	Bits	Type	Description
PC0, PC1	[7:4], [15:12]	rw	**Control for ESR Pin x** This bit field defines the ESR x functionality according to the coding tables (see **Table 8-12** and **Table 8-13**).
0	[3:0], [11:8], [31:16]	r	**Reserved** Read as 0; should be written with 0.

Pad Control Coding

Table 8-12 describes the coding of the PC0 bit field that determine the port line functionality.

Table 8-12 PC0 Coding

PC0[3:0]	I/O	Output Characteristics	Selected Pull-up/Pull-down/ Selected Output Function
$0X00_B$	Input is active and not inverted; Output is inactive		No input pull device connected
$0X01_B$			Input pull-down device connected
$0X10_B$			Input pull-up device connected
$0X11_B$			No input pull device connected
1000_B	Input is active and not inverted; Output is active	Push-pull	General-purpose Output
1001_B			Output drives a 0 for System Resets until ESROCFG.ARI is cleared, a weak pull-up is active otherwise
1010_B			Reserved, do not use this combination
1011_B			Reserved, do not use this combination
1100_B	the input is active and not inverted; Output is active	Open-drain	General-purpose Output
1101_B			Output drives a 0 for System Resets until ESROCFG.ARI is cleared , a weak pull-up is active otherwise
1110_B			Output drives a 0 for Application Resets until ESROCFG.ARI is cleared, a weak pull-up is active otherwise
1111_B			Reserved, do not use this combination

Pad Control Coding

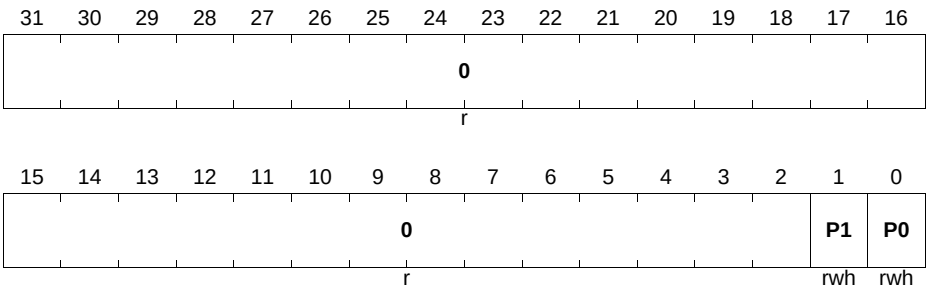
Table 8-13 describes the coding of the PC1 bit field that determine the port line functionality.

Table 8-13 PC1 Coding

PC1[3:0]	I/O	Output Characteristics	Selected Pull-up/Pull-down/ Selected Output Function
$0X00_B$	Input is active and not inverted; Output is inactive		No input pull device connected
$0X01_B$			Input pull-down device connected
$0X10_B$			Input pull-up device connected
$0X11_B$			No input pull device connected
1000_B	Input is active and not inverted; Output is active	Push-pull	General-purpose Output
1001_B			Reserved, do not use this combination
1010_B			Reserved, do not use this combination
1011_B			Reserved, do not use this combination
1100_B	the input is active and not inverted; Output is active	Open-drain	General-purpose Output
1101_B			Reserved, do not use this combination
1110_B			Output drives a 0 for Application Resets until ESROCFG.ARI is cleared, a 'Z' otherwise
1111_B			Reserved, do not use this combination

The output register determines the value of a GPIO pin when it is selected by IOCR as output. Writing a 0 to a OUT.Px (x = 0-1) bit position delivers a low level at the corresponding output pin. A high level is output when the corresponding bit is written with a 1. Note that each single bit or group of bits of OUT.Px can be set/cleared by writing appropriate values into the output modification register OMR.

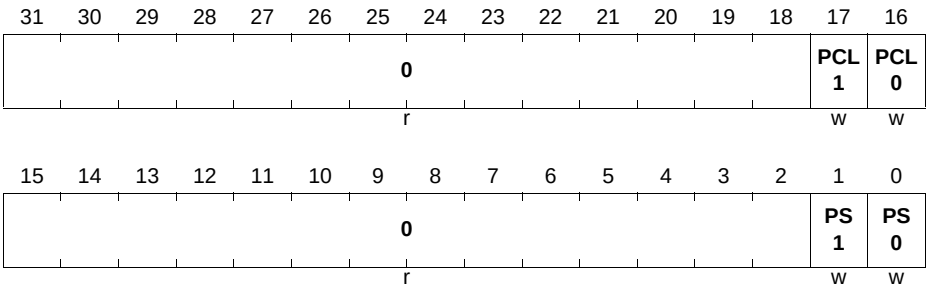
OUT
ESR Output Register **($0A4_H$)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
Px **(x = 0-1)**	x	rwh	**Output Bit x** This bit determines the level at the output pin $\overline{ESR}$x if the output is selected as GPIO output. 0_B The output level of $\overline{ESR}$x is 0 1_B The output level of $\overline{ESR}$x is 1 Px can also be set/cleared by control bits of the OMR register.
0	[31:2]	r	**Reserved** Read as 0; should be written with 0.

The output modification register contains control bits that make it possible to individually set, clear, or toggle the logic state of a single pad by manipulating the output register.

OMR
ESR Output Modification Register **($0A8_H$)** **Reset Value: 0000 0000$_H$**



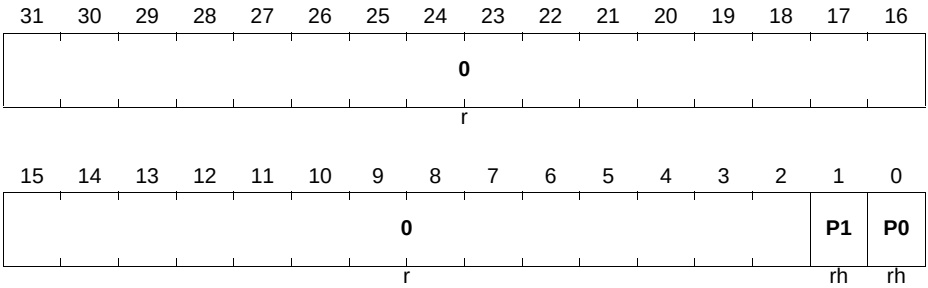
Field	Bits	Type	Description
PSx (x = 0-1)	x	w	**ESRx Pin Set Bit x** Setting this bit will set or toggle the corresponding bit in the output register OUT. The function of this bit is shown in **Table 8-14**. Reading this bit returns 0.
PCLx (x = 0-1)	x + 16	w	**ESRx Pin Clear Bit x** Setting this bit will clear or toggle the corresponding bit in the port output register OUT. The function of this bit is shown in **Table 8-14**. Reading this bit returns 0.
0	[15:2], [31:18]	r	**Reserved** Read as 0; should be written with 0.

Table 8-14 Function of the Bits PCLx and PSx

PCLx	PSx	Function
0	0	Bit OUT.Px is not changed
0	1	Bit OUT.Px is set
1	0	Bit OUT.Px is cleared
1	1	Bit OUT.Px is toggled

The logic level of a GPIO pin can be read via the read-only port input register IN. Reading the IN register always returns the current logical value at the GPIO pin independently whether the pin is selected as input or output.

IN
ESR Input Register **($0AC_H$)** **Reset Value: $0000\ 000X_H$**



Field	Bits	Type	Description
Px **(x = 0-1)**	x	rh	**Input Bit x** This bit indicates the level at the input pin $\overline{ESR}$x. 0_B The input level of ESRx is 0 1_B The input level of ESRx is 1
0	[31:2]	r	**Reserved** Read as 0.

Pad Disable Control Register

The pad structure of the TC21x/TC22x/TC23x GPIO lines offers the possibility to disable pad. This feature can be controlled by individual bits in the pad disable control register PDISC.

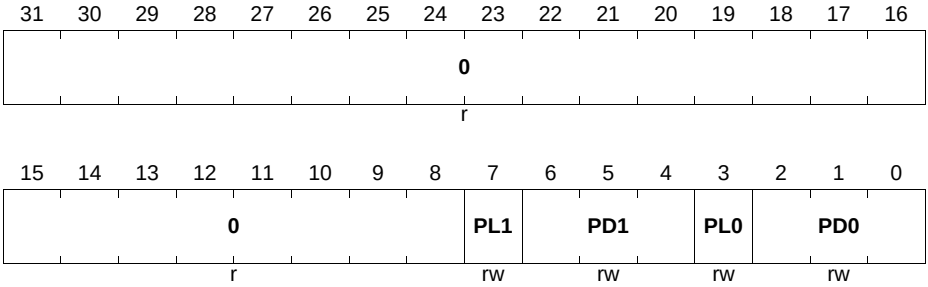
PDISC
Pad Disable Control Register **($18C_H$)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													**0**	**PDIS 1**	**PDIS 0**
r													rw	rw	rw

Field	Bits	Type	Description
PDISx **(x = 0-1)**	x	rw	**Pad Disable for ESR Pin x** This bit disables the pad. 0_B Pad Px is enabled 1_B Pad Px is disabled
0	[31:2]	r	**Reserved** Read as 0; should be written with 0.

PDR
ESR Pad Driver Mode Register (9C$_H$) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
PD0, PD1	[2:0], [6:4]	rw	**Pad Driver Mode for ESR Pins 0 and 1** PD0 selects the driver mode for ESR0 output. PD1 selects the driver mode for ESR1 output. X00$_B$ Speed Grade 1 X01$_B$ Speed Grade 2 X10$_B$ Speed Grade 3 X11$_B$ Speed Grade 4
PL0, PL1	3, 7	rw	**Reserved in this product** Read as 0 after reset; returns last value written
0	[31:8]	rw	**Reserved**

8.2.3 Boot Software Interface

In order to determine the correct starting point of operation for the software a minimum amount of hardware support is required. As much as possible is done via software. Some decisions have to be made in hardware because they must be known before any software is operational.

For a startup operation there are two general cases that have to be handled:

- Differentiation between Test Mode and Normal Mode for each Power-on Reset event (see **Section 8.2.3.1**)
- Configuration of the boot option for each Application Reset event (see **Section 8.2.3.2**)

8.2.3.1 Configuration done with Start-up

At device power-on some basic operating mode selection has to be done. The first decision that has to be made is whether the device should operate in Test Mode or in Normal (Customer) Mode. The Test Mode is only for Infineon internal device testing, it is not intended for any customer and is not related to debug.

If the Normal Mode was selected the next decision is which debug interface type issued for debugging for this session (until the next power-on event).

Table 8-15 Normal Mode / Test Mode Input Selection

Field	Description
TESTMODE	**Latched TESTMODE Signal** 0 A Test Mode can be selected 1 Normal Mode is selected
$\overline{\text{TRST}}$	**Latched $\overline{\text{TRST}}$ Signal** 0 The JTAG interface is active. 1 The DAP interface is active.

After these two decisions were made the detailed decision has to be made to define the real startup configuration. Most is made via the software and can be supported by some hardware selections depending on the startup configuration that should be selected.

8.2.3.2 Start-up Configuration Options

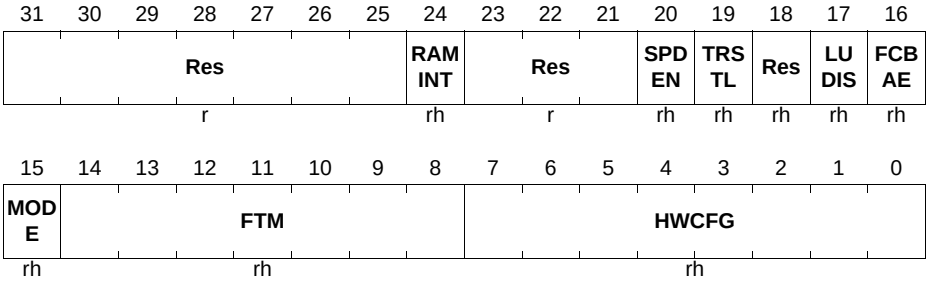
The states of the HWCFG port pin inputs are latched on the rising edge of an Application Reset and stored in register STSTAT.HWCFG. The update of bit field STSTAT.HWCFG with the latched value is only done if bit STSTAT.LUDIS is cleared. If bit STSTAT.LUDIS is set then the value of STSTAT.HWCFG is not updated.

8.2.3.3 Status Registers

Startup Status Registers

Register STSTAT contains the information used by the boot firmware to identify the start-up settings.

STSTAT
Start-up Status Register **($0C0_H$)** **Reset Value: $000X\ 80XX_H$**



Field	Bits	Type	Description
HWCFG	[7:0]	rh	**Hardware Configuration Setting** This bit field contains the value that is used by the boot software. This bit field is updated in case of an ApplicationReset with the content of the latches of the HWCFG port pins if bit STSTAT.LUDIS is cleared. This bit field is left unchanged in case of an ApplicationReset and is not updated with the content of the latches of the HWCFG port pins if bit RSTSTAT.SW are cleared and bit STSTAT.LUDIS is set. This bit field is updated with the value written to bit field STCON.HWCFG on a software write action. *Note: The observed reset value after boot depends upon the state of the HWCFG pins*

Field	Bits	Type	Description
FTM	[14:8]	rh	**Firmware Test Setting** In Normal Mode this bit field is updated with 0000000_B and should be ignored by the boot software.
MODE	15	rh	**MODE** This bit indicates if the Test Mode is entered or not. 0_B A Test Mode can be selected 1_B Normal Mode is selected
FCBAE	16	rh	**Flash Config. Sector Access Enable** 0_B Flash config sector is not accessible. Instead the flash memory area is accessed. 1_B Flash config sector is accessible. The flash memory area can not be accessed. This bit can be cleared by setting bit STCON.CFCBAE. This bit can be set by setting bit STCON.SFCBAE. *Note: Reset value of this bit is 0*
LUDIS	17	rh	**Latch Update Disable** 0_B Bit field STSTAT.HWCFG is automatically updated with the latched value of the HWCFG input pins 1_B Bit field STSTAT.HWCFG is not updated with the latched value of the HWCFG input pins This bit can be set by setting bit SYSCON.SETLUDIS. *Note: Reset value of this bit is 0*
Res	18	rh	**Reserved** **Read as 0; should be written with 0.**
TRSTL	19	rh	**TRSTL Status** This bit simply displays the value of TRSTL.
SPDEN	20	rh	**Single Pin DAP Mode Enable** 0_B Single Pin DAP Mode is disabled 1_B Single Pin DAP Mode is enabled
Res	[23:21]	r	**Reserved** Read as 0; should be written with 0.

Field	Bits	Type	Description
RAMINT	24	rh	**RAM Content Security Integrity** In normal operation this bit can be set or cleared by the application (via SYSCON). If a test boot mode is entered, the bit is automatically cleared (and cannot be set again in test mode) because the content may have been altered 0_B RAM Security Integrity cannot be guaranteed 1_B RAM Security Integrity maintained Note: This bit is reset only by a cold power-on reset.
Res	[31:25]	r	**Reserved** Read as 0; should be written with 0.

8.2.4 NMI Trap Generation

The NMI trap structure is shown in **Figure 8-20**. The trap request trigger or the corresponding trap set bit (in register TRAPSET) can trigger the NMI trap generation. An NMI trap is always issued simultaneously to all CPUs in the system. The trap flag can be cleared by software by writing to the corresponding bit in register TRAPCLR. A NMI request is only generated if the trap source was not disabled. Otherwise only the trap status flag is set but no NMI request is generated.

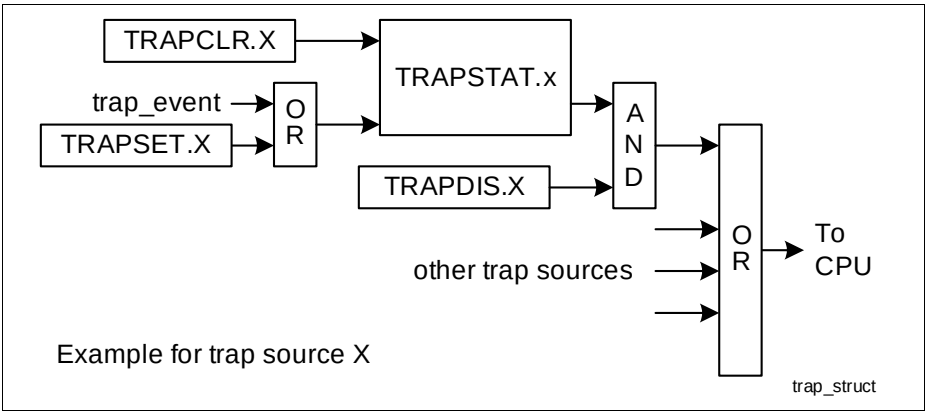


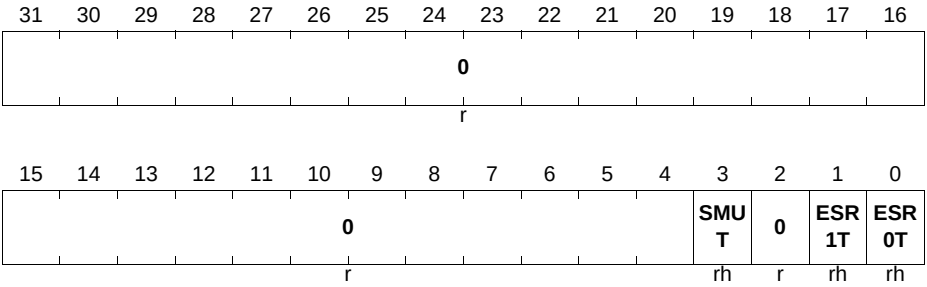
Figure 8-20 NMI Trap Generation

Handling NMI Traps

As an NMI trap is generated while the trap source is enabled AND the trap status flag is set, it is recommended to clear the trap status flag before the trap source is enabled. The trap status flag can be set before the trap source is enabled and simply enabling the trap source can result in unintended NMI traps. At the end of a NMI trap handling routine the trap status flag should be cleared.

8.2.4.1 Trap Control Registers

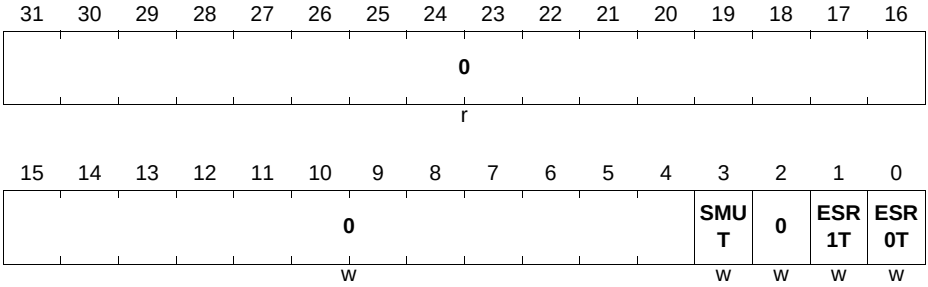
TRAPSTAT
Trap Status Register **(124_H)** **Reset Value: 0000 000X$_H$**



Field	Bits	Type	Description
ESR0T	0	rh	**ESR0 Trap Request Flag** This bit is set if an $\overline{\text{ESR0}}$ event is triggered. 0_B No trap was requested since this bit was cleared the last time 1_B A trap was requested since this bit was cleared the last time This bit can be cleared by setting bit TRAPCLR.ESR0T. This bit can be set by setting bit TRAPSET.ESR0T. *Note: Observed reset value after boot will depend upon ARI mode because of ESR pin transition, see **Table 8-11***
ESR1T	1	rh	**ESR1 Trap Request Flag** This bit is set if an $\overline{\text{ESR1}}$ event is triggered. 0_B No trap was requested since this bit was cleared the last time 1_B A trap was requested since this bit was cleared the last time This bit can be cleared by setting bit TRAPCLR.ESR1T. This bit can be set by setting bit TRAPSET.ESR1T. *Note: Reset value of this bit is 0*

Field	Bits	Type	Description
SMUT	3	rh	**SMU Alarm Trap Request Flag** This bit is set if an SMU Alarm is indicated . 0_B No trap was requested since this bit was cleared the last time 1_B A trap was requested since this bit was cleared the last time This bit can be cleared by setting bit TRAPCLR.SMUT. This bit can be set by setting bit TRAPSET.SMUT. *Note: Reset value of this bit is 0*
0	2, [31:4]	r	**Reserved** Read as 0.

TRAPSET
Trap Set Register **(128_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
ESR0T	0	w	**Set Trap Request Flag ESR0T** Setting this bit sets bit TRAPSTAT.ESR0T. Clearing this bit has no effect. Reading this bit returns always zero.
ESR1T	1	w	**Set Trap Request Flag ESR1T** Setting this bit sets bit TRAPSTAT.ESR1T. Clearing this bit has no effect. Reading this bit returns always zero.

Field	Bits	Type	Description
SMUT	3	w	**Set Trap Request Flag SMUT** Setting this bit sets bit TRAPSTAT.SMUT. Clearing this bit has no effect. Reading this bit returns always zero.
0	2, [31:4]	r	**Reserved** Read as 0; should be written with 0.

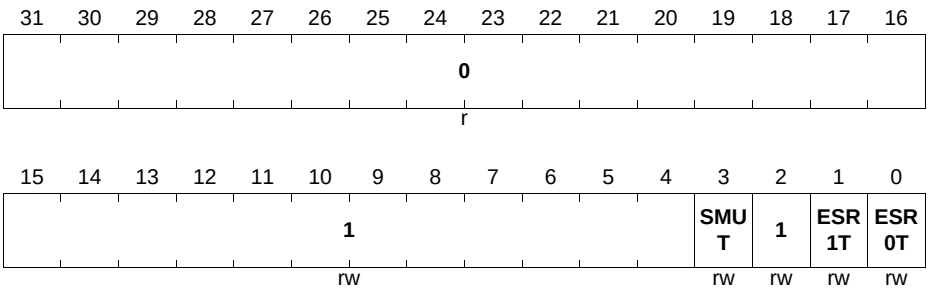
TRAPCLR
Trap Clear Register **(12C$_H$)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												**SMUT**	**0**	**ESR1T**	**ESR0T**
w												w	w	w	w

Field	Bits	Type	Description
ESR0T	0	w	**Clear Trap Request Flag ESR0T** Setting this bit clears bit TRAPSTAT.ESR0T. Clearing this bit has no effect. Reading this bit returns always zero.
ESR1T	1	w	**Clear Trap Request Flag ESR1T** Setting this bit clears bit TRAPSTAT.ESR1T. Clearing this bit has no effect. Reading this bit returns always zero.
SMUT	3	w	**Clear Trap Request Flag SMUT** Setting this bit clears bit TRAPSTAT.SMUT. Clearing this bit has no effect. Reading this bit returns always zero.
0	[31:4], 2	r	**Reserved** Read as 0; should be written with 0.

TRAPDIS
Trap Disable Register **(130_H)** **Reset Value: 0000 FFFF$_H$**



Field	Bits	Type	Description
ESR0T	0	rw	**Disable Trap Request ESR0T** 0_B A trap request can be generated for this source 1_B No trap request can be generated for this source
ESR1T	1	rw	**Disable Trap Request ESR1T** 0_B A trap request can be generated for this source 1_B No trap request can be generated for this source
SMUT	3	rw	**Disable Trap Request SMUT** 0_B A trap request can be generated for this source 1_B No trap request can be generated for this source
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.
1	[15:4], 2	r	**Reserved** Read as 1; should be written with 1.

8.2.5 RCU Register Address

Table 8-16 Registers Address Spaces - RCU Kernel Registers

Module	Base Address	End Address	Note
SCU	$F003\ 6000_H$	$F003\ 63FF_H$	-

8.2.6 RCU Kernel Registers

This section describes the kernel registers of the RCU module. Most of the RCU kernel register names described in this section will be referenced in other parts of the TC21x/TC22x/TC23x User's Manual by the module name prefix "SCU_".

RCU Kernel Register Overview

Table 8-17 Register Overview of RCU (Offset from SCU Register Base)

Short Name	Long Name	Offset Addr. [1]	Access Mode		Reset	Description See
			Read	Write		
RSTSTAT	Reset Status Register	050_H	U, SV	BE	Power-on Reset	Page 8-92
–	Reserved	054_H	BE	BE	–	–
RSTCON	Reset CON Register	058_H	U, SV	SV, SE, P	Power-on Reset	Page 8-95
ARSTDIS	Application Reset Disable Register	$05C_H$	U, SV	SV, SE, P	Power-on Reset	Page 8-97
SWRSTCON	Software Reset Configuration Register	060_H	U, SV	SV, SE, P	Power-on Reset	Page 8-99
RSTCON2	Reset Configuration Register 2	064_H	U, SV	SV, SE, P	Cold Power-on Reset	Page 8-100
Reserved	Reserved	068_H	U, SV	BE	-	-
ESRCFG0	ESR0 Configuration Register	070_H	U, SV	SV, SE, P	System Reset	Page 8-105

Table 8-17 Register Overview of RCU (Offset from SCU Register Base)

Short Name	Long Name	Offset Addr. 1)	Access Mode		Reset	Description See
			Read	Write		
ESRCFG1	ESR1 Configuration Register	074_H	U, SV	SV, SE, P	System Reset	Page 8-105
ESROCFG	ESR Reset Output Configuration Register	078_H	U, SV	SV, SE, P	System Reset	Page 8-106
PDR	Pad Driver Mode Register	$09C_H$	U, SV	SV, E, P	System Reset	Page 8-113
IOCR	Input/Output Control Register	$0A0_H$	U, SV	U, SV, P	System Reset	Page 8-107
OUT	Output Register	$0A4_H$	U, SV	U, SV, P	System Reset	Page 8-110
OMR	Output Modification Register	$0A8_H$	U, SV	U, SV, P	System Reset	Page 8-110
IN	Input Register	$0AC_H$	U, SV	BE	System Reset	Page 8-111
STSTAT	Start-up Status Register	$0C0_H$	U, SV	BE	Power-on Reset (RAMINT on cold Power-on only)	Page 8-115
Reserved	Reserved	$0C4_H$	U, SV	BE	Power-on Reset	
TRAPSTAT	Trap Status Register	124_H	U, SV	BE	System Reset	Page 8-119
TRAPSET	Trap Set Register	128_H	U, SV	SV, E, P	System Reset	Page 8-120
TRAPCLR	Trap Clear Register	$12C_H$	U, SV	U, SV, P	System Reset	Page 8-121
TRAPDIS	Trap Disable Register	130_H	U, SV	SV, E, P	Application Reset	Page 8-122

1) The absolute register address is calculated as follows:
Module Base Address + Offset Address (shown in this column)

8.3 Power Supply & Power Management Controller (PMC)

This chapter describes the Power Supply of the TC21x/TC22x/TC23x and the Power Management Controller (PMC).

This is described in the following sections:

- Power Supply and Control (see **Section 8.3.1**)
- Power Management (see **Section 8.3.2**)
- Power Supply and Power Management Registers (see **Table 8-24**)

8.3.1 Power Supply and Control

8.3.1.1 Introduction

On-chip linear and switch mode voltage regulators are implemented in TC21x/TC22x/TC23x thereby enabling a single source power supply concept. The external nominal system supply is 3.3 V. The Embedded Voltage Regulator (EVR13) in turn generate the 1.3 V supply required internally for the core, flash and port domains. Linear Drop Out (LDO) or Switch Mode Power Supply (SMPS) mode may be selected for the EVR13 regulator.

Depending on the chosen EVR mode, the actual power consumption, EMC requirements and thermal constraints of the system; additional external components may be required. It is also possible to supply the 1.3 V supply voltage externally thus ensuring compliance to the legacy supply concept.

All supply and generated voltages are monitored internally against overshoot and brownout conditions based on programmable thresholds. In case these thresholds are violated, either a cold power on reset is triggered or an alarm is provided to the SMU.

All internal supplies except analog domain (V_{AREF} & V_{DDM}) are supplied by the EVRs. The analog supply domain is separated from the main EVR supply domain and can be supplied by separate external regulators or trackers. It is possible to have a unified supply scheme with a 3.3V ADC domain (V_{DDM} / V_{AREFx} = 3.3 V) and the remaining system also running on 3.3 V supply (V_{DDP3} = 3.3 V). Alternatively it is also possible to have a mixed supply scheme with a 5 V ADC domain (V_{DDM} / V_{AREFx} = 5 V) and the remaining system running on 3.3 V supply (V_{DDP3} = 3.3 V).

8.3.1.2 Supply Mode and Topology Selection

The choice of the supply scheme at startup is based on the latched status of HWCFG[0] and HWCFG[2] pins before PORST release and is indicated by **PMSWSTAT**.HWCFGEVR [0,2] status flags. Following supply modes are supported and are further enumerated in **Table 8-18**.

- Single source 3.3 V supply level is supported in following topologies.
  - EVR13 in SMPS mode with external switch capacitors. (TC23x only)
  - EVR13 in LDO mode with internal pass devices. (TC23x & TC22x)
- Supplies are provided externally and the EVR is in disabled state.
  - 3.3 V and 1.3 V are supplied externally.(TC23x & TC22x)

EVR13 is enabled or disabled at startup via the HWCFG[2] configuration pin. The selection between LDO or SMPS topology for EVR13 is decided via HWCFG[0] configuration pin. In case EVR13 SMPS mode is selected, V_{CAP0} and V_{CAP1} pins shall be connected to an external switch capacitor as shown **Figure 8-24**. In case EVR13 LDO mode is selected, V_{CAPx} pin shall be left unconnected.

In case of smaller devices and packages, HWCFG[0,2] configuration pins may be absent and the EVR is active at startup in LDO mode with internal pass devices. In case of TC22x, SC SMPS EVR13 regulator is absent and 1.3 V supply may only be generated using internal LDO pass devices. EVR13 may be disabled via **EVR13CON**.EVR13OFF bit if required. The current state of EVR13 are reflected in **EVRSTAT**.EVR13 flags.All supplies must be available and be stable with the release of the PORST signal.

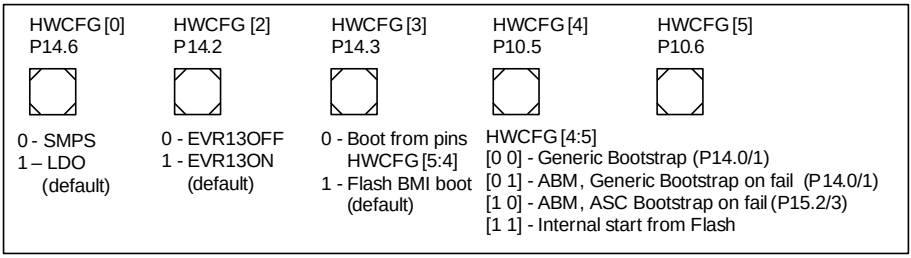
Table 8-18 Supply Mode and Topology selection

No.	HWCFG [0,2][1)]	VCAP0 VCAP1[2)]	Supply Pin Voltage Level / Source[3)]	Selected Supply Scheme
e.)	$0,1_B$	V_{CAP0}/V_{CAP1} pins connected to external flying capacitor.	V_{DDP3}= 3.3V V_{DDM}/V_{AREFx} = 5V / 3.3V V_{DD} = EVR13 SMPS $V_{SS}/V_{SSM}/V_{AGND}$ = 0 V	3.3 V single source supply. EVR13 in SMPS mode with external switch capacitor. 5 V or 3.3 V ADC domain. Standby Mode supported.
f.)	$1,1_B$	V_{CAP0}/V_{CAP1} pins may be left open.	V_{DDP3}= 3.3V V_{DDM}/V_{AREFx} = 5V / 3.3V V_{DD} = EVR13 LDO $V_{SS}/V_{SSM}/V_{AGND}$ = 0 V	3.3 V single source supply. EVR13 in LDO mode with internal pass devices. 5 V or 3.3 V ADC domain. Standby Mode supported.
h.)	$X,0_B$	V_{CAP0}/V_{CAP1} pins shall be left open.	V_{DDP3}= 3.3V V_{DDM}/V_{AREFx} = 5V / 3.3V V_{DD} = 1.3 V $V_{SS}/V_{SSM}/V_{AGND}$ = 0 V	3.3 V and 1.3 V are supplied externally. EVR13 inactive. 5 V or 3.3 V ADC domain. Standby Mode supported and 1.3V supply shall be switched off by external regulator after Standby state is entered.

1) HWCFG[0,2] pins are configured as inputs with weak pull-up after reset, therefore if pins are left unconnected, it is ensured that EVR13 LDO is active by default. These pads have default pull-up during reset unlike the normal port pads which are predominantly in tristate during reset.

2) VCAPx pins are dedicated for SMPS Switch capacitor SMPS mode and cannot be used as port pins, In SMPS mode, the VCAPx shall be connected with a flying capacitor between the pins. Incase of LDO mode, the VCAP pins may be left open or may be connected with a small decoupling capacitor. VCAP pins shall not be connected to ground as ESD diodes start conducting.

3) It shall be ensured that all VDD pins are connected together on PCB level. Likewise It shall be ensured that all VDDP3 pins are connected together on PCB level.



1.) In QFP100 and smaller packages, HWCFG [2] pin is absent and pulled high ensuring EVR13 is active.

2.) Incase HWCFG pins are left unconnected, EVR13 LDO mode is selected owing to default pull-up state.

3.) HWCFG [0:2] pins are latched during supply ramp-up (VDDP3 < 2.97V) and stored in PMSWSTAT.HWCFGEVR & TRIST register bits. The remaining HWCFG pins are latched on internal reset release (between 100us – 180us after reset assertion) and stored in STSTAT register.

Figure 8-21 Hardware Configuration (HWCFG) pins

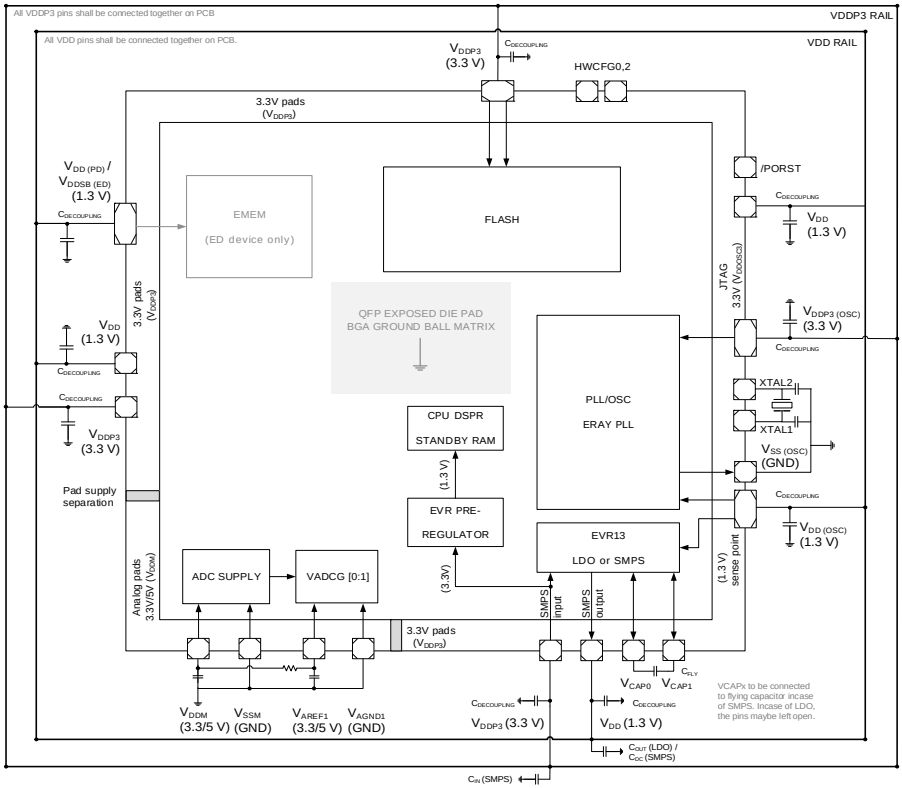


Figure 8-22 Supply Routing

8.3.1.3 Linear Regulator Mode

The LDO part of the EVR is composed of

- Infrastructure components : EVR internal pre-regulator, bandgaps and clock sources
- EVR13 main LDO regulator

The EVR13 regulator supplies mainly the core domain including RAMs. The EVR constitutes a digital regulator, a pass device control unit and a voltage feedback loop. The pass device outputs are appropriately buffered by capacitors to handle load transients so as not to violate the operating voltage limits. The EVR can be individually disabled via HWCFG[2] pin as described in **Chapter 8.3.1.2**. The EVR13 can also be disabled via **EVR13CON**.EVR13OFF protected register bit .

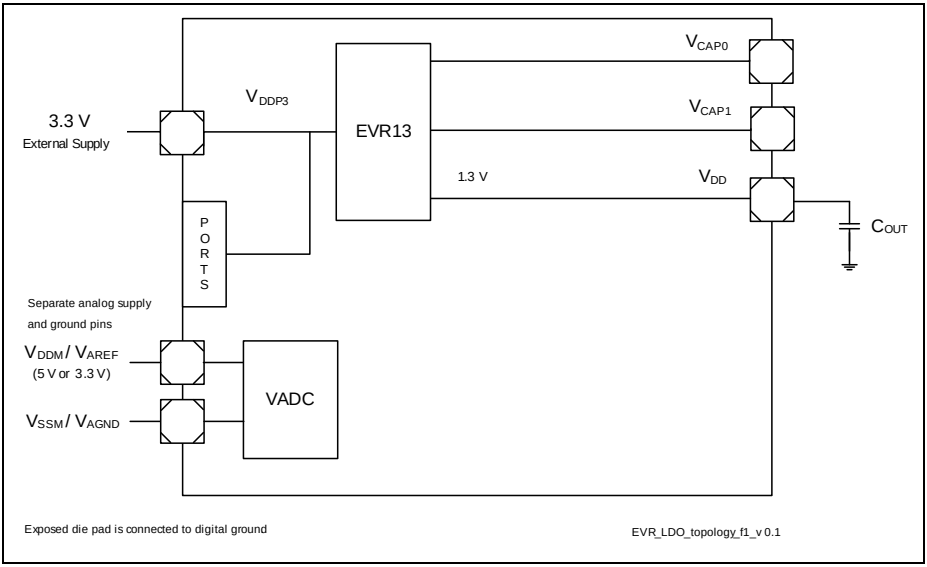


Figure 8-23 EVR LDO topology (f) - 3.3 V single supply with int. pass device

8.3.1.4 Switch Capacitor Regulator Mode

The **S**witch **C**apacitor SMPS regulator provides a higher efficiency of power conversion compared to the linear voltage regulator concept. However it requires additional external components and injects more switching noise into the system. The integrated switch capacitor regulator modulates internal switches to buffer the energy in capacitors in order to generate a regulated core supply. A flying capacitor and a buffer capacitor is required as shown in **Figure 8-24** .

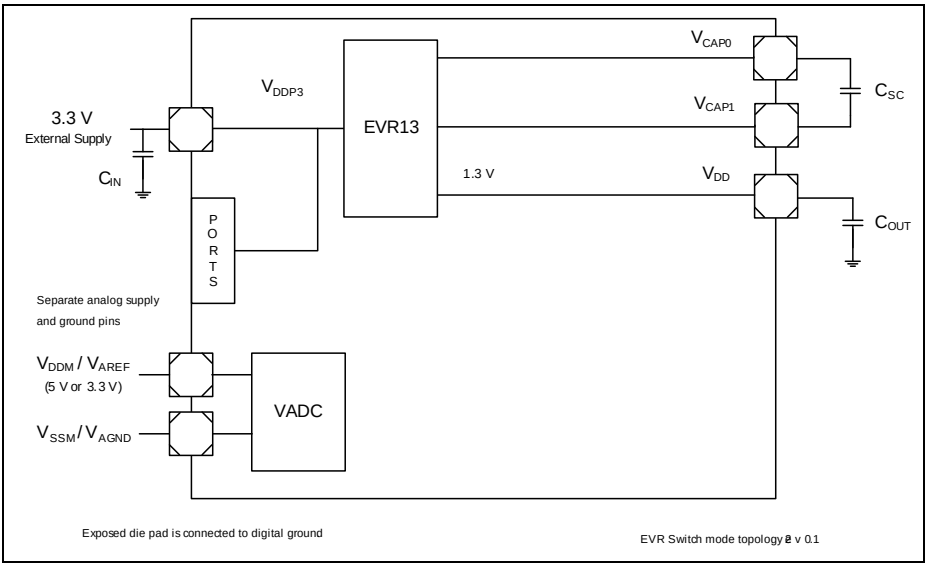


Figure 8-24 EVR Switch mode topology (e) - 3.3 V single supply

The control strategy involves synchronous switching of the 4 switches at a defined switching frequency in conductance modulation mode thus charging and discharging the switch capacitor network. The charge and discharge time are programmable via **EVRSDCTRL1**.TON and TOFF fields which inturn determine the switching frequency. The output voltage is measured via the SMPS ADC module and based on the deviation to the reference voltage the controller modulates the conductance value for the next charge / discharge period. The nominal efficiency of conversion is approximately 75 %. The efficiency of conversion varies within 70 % - 85 % as the input supply varies within the V_{DDP3} +- 10% operation range.

During the start-up phase, a soft-startup control strategy is used in order to avoid an overshoot of the output voltage and the charging of the buffer capacitor takes place gradually. During the initial switching periods a different set of coefficients are used. The start phase ends when the ADC indicates that the output voltage has reached the target value. At this point the PI controller is re-configured and the normal closed loop operation begins. After a voltage transient (typically an overshoot), EVR13 is ready. The step-down regulator and the associated bandgaps are trimmed later to achieve a more accurate output voltage.

In case frequency spreading is activated, the charge and discharge times are appended with a uniform random offset once per switching period. This allows to randomize the switching frequency of the SC DCDC regulator within a maximum value. The maximum random offset can be programmed via **EVRSDCTRL1**.SDFREQSPRD register. It is

possible to synchronize an external step-down regulator which may run at much lower switching frequencies than the internal regulator. A scaled synchronisation clock output is provided and routed to an external pin (P33.11 pin - DCDCSYNC output).

In case the current consumption in SC SMPS mode exceeds the switching regulator capabilities as indicated in the datasheet and the maximum conductance limit is reached, a switch of the switch capacitor mode takes place from 2 : 1 to 1 : 1 mode. A consequent interrupt (SRC_EVRSCDC) is issued to the system. The switch from the SC 2:1 mode (SCMOD = 01b) to SC 1:1 mode (SCMOD = 00b) is also indicated via **EVRSTAT**.SCMOD status bits. The lower bit of **EVRSTAT**.SCMOD[12] signal is used to generate the interrupt on SRC_EVRSCDC interrupt node on a transition from 1 to 0.

It shall be ensured that all the VDD pins are connected together when using SC DCDC regulator. The VDD output pin adjacent to the VCAP pins is separated from the VDD feedback sense pin as shown **Figure 8-22** and needed to be connected together for proper functioning of regulator closed loop.

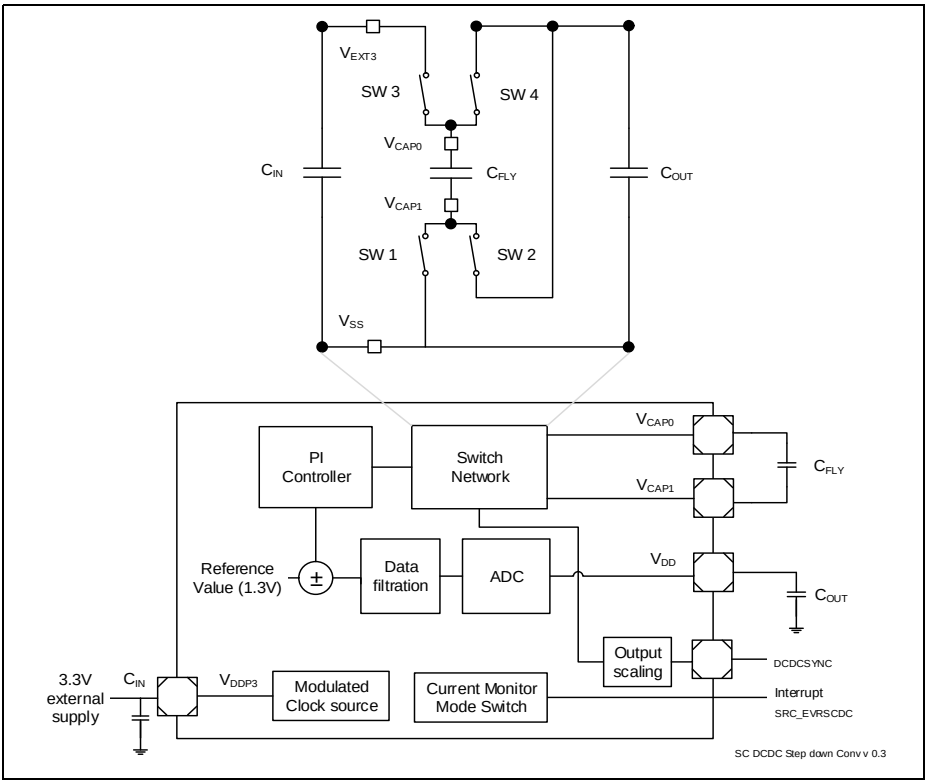


Figure 8-25 Switch Capacitor SMPS regulator

8.3.1.5 External Supply Mode

The external supply mode involves deactivating EVR13 and providing both 3.3 V and 1.3 V supply externally. In this mode, EVR13 is disabled via the HWCFG[2] configuration pin .

Following external supply modes are supported.

- 3.3 V and 1.3 V supplied externally as shown in **Figure 8-26** .

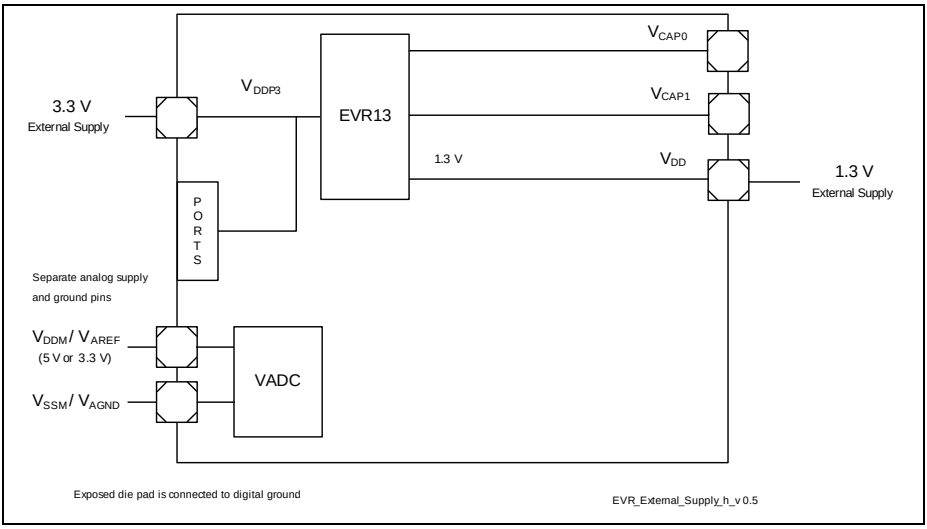


Figure 8-26 External Supply mode (h) - 3.3 V and 1.3 V externally supplied

8.3.1.6 Components and Layout

The efficiency of EVR is influenced by the characteristics of the selected components and also the placement and routing of the components on the PCB. In case of EVR13 LDO regulator, an appropriate buffer capacitor is required at the 1.3 V V_{DD} regulator supply output to handle load and line jump and regulation requirements based on application needs. The additional external components for the SC SMPS regulator constitute a flying capacitor (1 uF) and a buffer capacitor (10 uF). An input capacitor (4,7 uF) is required in case of SC SMPS mode to limit the input current ripple. The trace impedances and distances to the external components should be in principle as small as possible. The component requirements are documented in the datasheet and recommendations are also provided in application notes.

In case of usage of Emulation devices, it should be taken care that the component choice also considers the current additionally drawn by ED RAM and EEC part.

8.3.1.7 Voltage Monitoring

The PMC module implements a staggered voltage monitoring build upon a primary and a secondary monitor. The primary monitor ensures that the microcontroller is put into a reset state when the lowest operational threshold is violated. The secondary monitor serves as an additional safety monitor providing over- and under-voltage alarms.

Primary under-voltage monitoring of the external V_{DDP3} supply and V_{DD} / EVR13 supplies are inherently carried out to ensure proper functioning of the system. The EVR control loop as well as the under-voltage monitors are based on a primary bandgap reference. The thresholds for the basic monitoring are non-configurable and represents the lowest possible thresholds for the correct functioning of the system. In case of violation of these thresholds, cold $\overline{\text{PORST}}$ is activated and $\overline{\text{PORST}}$ pin is pulled low (strong current sink). Following the reset release and firmware boot, it can be inferred from STBYR, EVR13 or SWD bits in RSTSTAT register as to whether the violation of these thresholds led to the previous reset. The primary under-voltage monitoring is kept active even if the respective EVR has been disabled and the supply is provided externally as shown in **Table 8-19**.

The primary Supply WatchDog (SWD monitor) monitors the ramp-up of external supply voltage and keeps the microcontroller in cold Power On Reset state as long as the supply has not reached the operational range. Likewise, it also detects ramp-down or brown out condition of external supply and sets the device into a cold Power On Reset state when the voltage has dropped below the lowest operational threshold. Nevertheless, It is recommended to monitor externally all supplies generated external to the microcontroller and to assert $\overline{\text{PORST}}$ reset pin in case of violation of the lowest operational limits.

The violation of the primary under-voltage thresholds of EVR13 and EVR33 is communicated to the HSM module. HSM module may lock access to EVR registers via SLCK bit so that supply generation cannot be influenced by other masters. This is to ensure that trojan programs do not manipulate the supplies to gain access to the system.

The external V_{DDP3} supply and V_{DD} / EVR13 supplies are measured by 8 bit analog converters and the measured value is indicated in **EVRADCSTAT** register. This may be used to do a plausibility check of external regulator supplies during runtime.To support safety startup tests including the EVR monitoring mechanism, it shall be possible to measure the EVR HPBG bandgap voltage (G0CH12), V_{DDP3} /2 (G1CH13) and V_{DD} / EVR13 (G1CH12) supplies using VADC converter channels. This may be used to also to do a plausibility check of the internal EVR bandgap voltage against the external VAREF / VAGND reference supplies.

Additional secondary over-voltage and under-voltage monitoring against programmable thresholds is provided for all supplied and generated voltages using a secondary bandgap reference. This includes the external supply voltage and the internally generated EVR13 output voltages as shown in **Figure 8-28**. The secondary voltage monitors are kept active even if the respective EVRs have been disabled and the supply is provided externally as shown in **Table 8-19**. In case of a threshold violation, an SMU alarm event is generated.

The secondary monitor violation is notified depending on the direction of voltage transition as programmed in **EVRMONCTRL** register. The appropriate thresholds for voltage monitoring can be programmed in the **EVROVMON** and **EVRUVMON** registers. In case of an active monitoring violation, respective status flags are set in the **EVRSTAT** register. It can be inferred from OV13 or OVSWD bits in **EVRSTAT** register as to whether over-voltage thresholds for the respective voltage domains were violated. Likewise, It can be inferred from UV13 or UVSWD bits in **EVRSTAT** register as to whether under-voltage thresholds were violated. It is also possible to deactivate the generation of SMU alarms from secondary monitors in **EVRMONCTRL** register.

The primary bandgap is compared with a secondary bandgap to detect bandgap drifts as reflected in **EVRSTAT**.BGPROK register bit. The bandgap BIST is only carried out during a supply ramp-up.

In case of primary monitor violation, respective status bits are set to indicate the event as shown in **Table 8-19**. These bits maybe evaluated during consequent start-up after cold power-fail reset to recognize which among the supply rails had the power-fail. Likewise for secondary monitor violation, the respective status bits may be evaluated to discriminate between an overvoltage or an undervoltage event and to recognize which supply rail had triggered the alarm event to SMU.

Table 8-19 Voltage Monitoring

Supply Pin / Rail	Primary Under-voltage monitor State (ON/OFF)	Secondary Over & Under-voltage Monitor State (ON/OFF)	Supply Range V	Is the Pin supplied
RUN or SLEEP system mode during supply modes e-h.				
V_{DDP3}	ON. RSTSTAT.SWD RSTSTAT.STBYR	ON **EVRSTAT**.OVSWD **EVRSTAT**.UVSWD	2.97–3.63 V	External 3.3V Supply to be provided
V_{DDM}	-	-	2.97–5.50 V	External Supply to be provided
V_{DD}	ON RSTSTAT.EVR13	ON **EVRSTAT**.OV13 **EVRSTAT**.UV13	1.17–1.43 V	EVR13 active or external 1.3V supply to be provided
STANDBY system mode during supply modes e-h.				
V_{DDP3}	ON RSTSTAT.STBYR	OFF	2.97–3.63 V	External 3.3V Supply to be provided
V_{DDM}	-	-	0–5.50 V	ON or OFF
V_{DD}	OFF	OFF	0 V	OFF

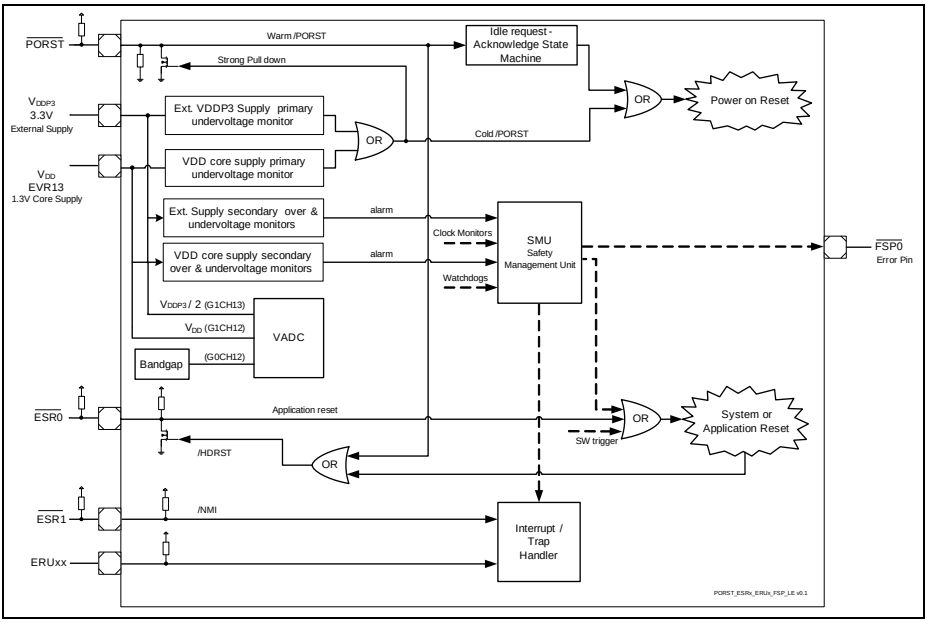
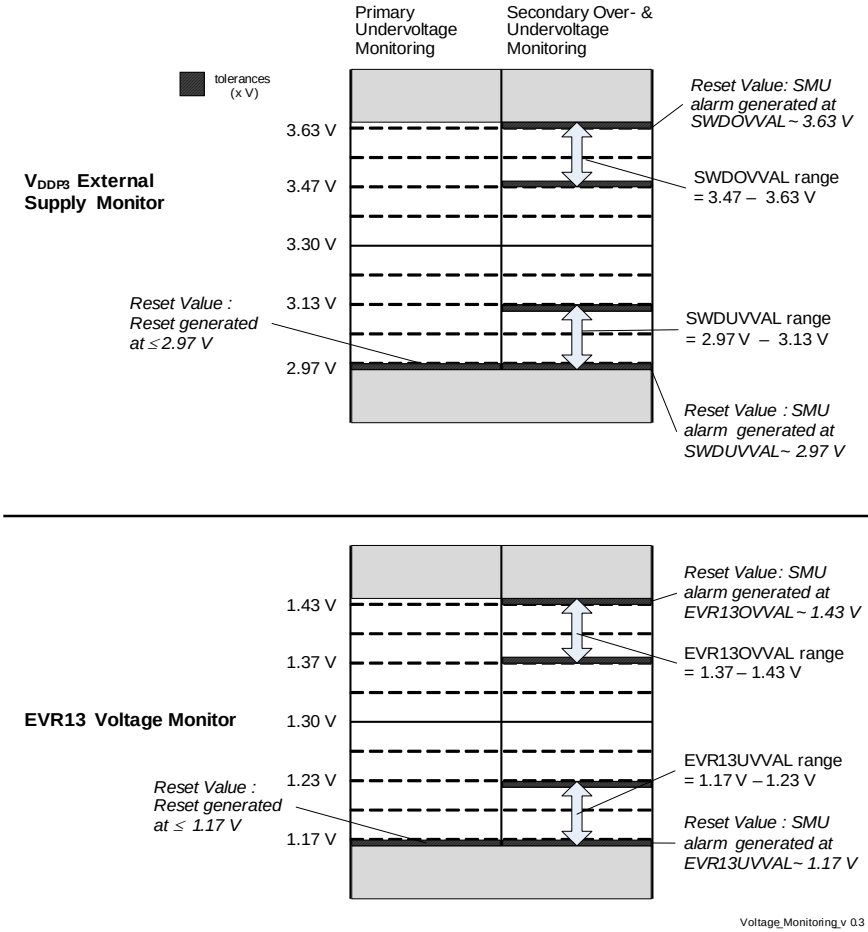


Figure 8-27 Monitoring and Reset



Monitoring tolerances and levels are specified in datasheet section EVR : Supply monitoring

Figure 8-28 Voltage Monitoring

8.3.1.8 Sequence during Power-up and Power-down

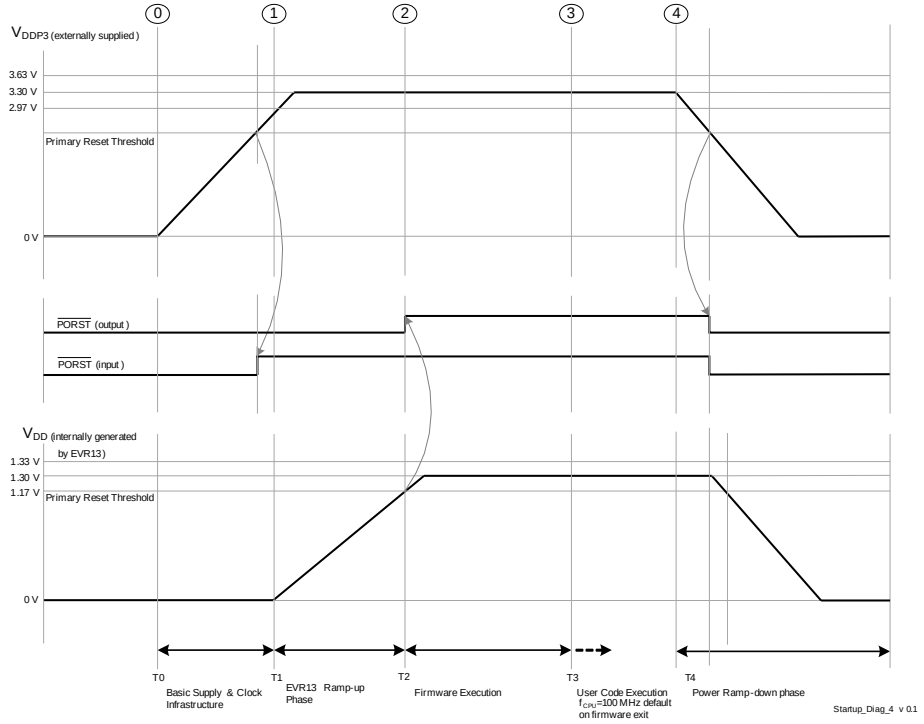


Figure 8-29 Single Supply mode (e & f) - 3.3 V single supply

Single Supply mode (e & f)

3.3 V single supply mode. 1.3 V is generated internally by the EVR13 regulator.

- The rate at which current is drawn from the external regulator (dI_{DDP3} /dt) is limited during the basic infrastructure and EVR13 regulator start-up phase (T0 upto T2) to a maximum of 100 mA/100 us. EVR13 is also robust against a voltage ramp-up starting from a residual voltage between 0 - 1 V. Start-up slew rates for supply rails should comply to datasheet values.
- Furthermore it is also ensured that the current drawn from the external regulator (dI_{DDP3} /dt) is limited during the Firmware start-up phase (T2 upto T3) to a maximum of 100 mA/100 us.
- PORST is active/ asserted when either PORST (input) or PORST (output) is active/ asserted.
- PORST (input) active means that the reset is held active by external agents by pulling the PORST pin low. It is recommended to keep the PORST (input) asserted until the external supply is above the respective primary reset threshold.
- PORST (output) active means that µC asserts the reset internally and drives the PORST pin low thus propagating the reset to external devices. The PORST (output) is asserted by the µC when atleast one among the two supply domains (1.3 V or 3.3 V) violate their primary under-voltage reset thresholds.The PORST (output) is deasserted by the µC when all supplies are above their primary reset thresholds and the basic supply and clock infrastructure is available.
- The power sequence as shown in **Figure 8-29** is enumerated below
  - T1 refers to the point in time when basic supply and clock infrastructure is available as the external supply ramps up. The supply mode is evaluated based on the HWCFG[0,2] pins and consequently a soft start of EVR13 regulator is initiated.
  - T2 refers to the point in time when all supplies are above their primary reset thresholds. EVR13 regulator has ramped up. PORST (output) is deasserted and HWCFG[3:5] pins are latched on PORST rising edge. Firmware execution is initiated.
  - T3 refers to the point in time when Firmware execution is completed. User code execution starts with a default frequency of 100 MHz.
  - T4 refers to the point in time during the Ramp-down phase when atleast one of the externally provided or generated supplies (1.3 V or 3.3 V) drop below their respective primary under-voltage reset thresholds.

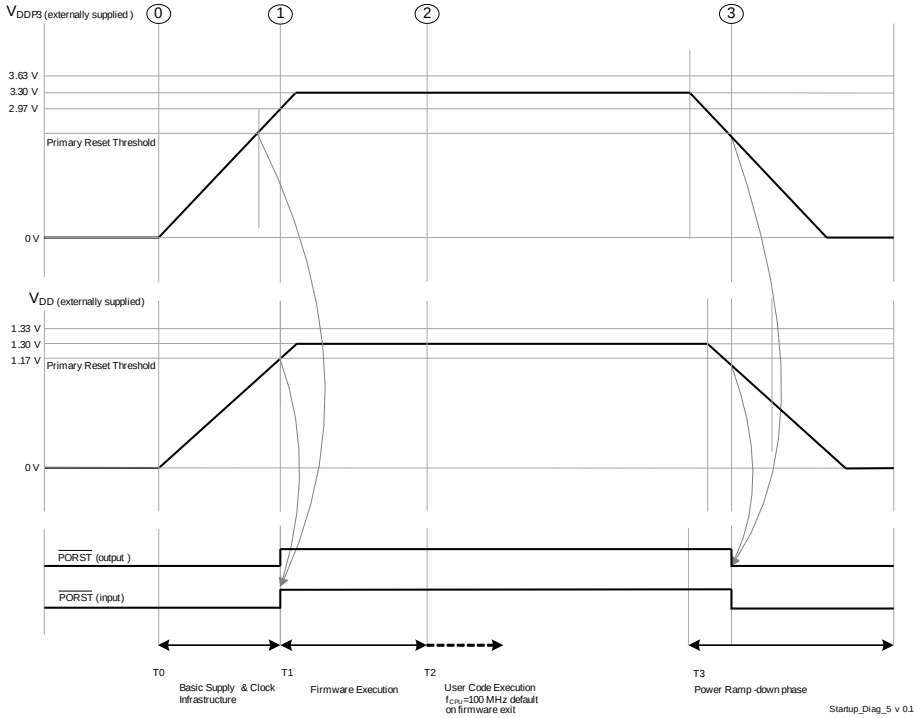


Figure 8-30 External Supply mode (h) - 3.3 V and 1.3 V external supply

External Supply mode (h)

All supplies, namely 3.3 V & 1.3 V, are externally supplied.

- External supplies V_{DDP3} & V_{DD} may ramp-up or ramp-down independent of each other with regards to start, rise and fall time(s). The supply system is also robust against a voltage ramp-up starting from a residual voltage between 0 - 1 V. Start-up slew rates for supply rails should comply to datasheet values.
- The rate at which current is drawn from the external regulator (dI_{DDP3} /dt, dI_{DD} /dt) is limited in the Start-up phase to a maximum of 100 mA/100 us.
- PORST is active/ asserted when either PORST (input) or PORST (output) is active/ asserted.
- PORST (input) active means that the reset is held active by external agents by pulling the PORST pin low. It is recommended to keep the PORST (input) asserted until all the external supplies are above their primary reset thresholds.
- PORST (output) active means that µC asserts the reset internally and drives the PORST pin low thus propagating the reset to external devices. The PORST (output) is asserted by the µC when atleast one among the two supply domains (1.3 V or 3.3 V) violate their primary under-voltage reset thresholds.The PORST (output) is deasserted by the µC when all supplies are above their primary reset thresholds and the basic supply and clock infrastructure is available.
- The power sequence as shown in **Figure 8-30** is enumerated below
  - T1 refers to the point in time when all supplies are above their primary reset thresholds and basic clock infrastructure is available. The supply mode is evaluated based on the HWCFG[0,2] pins. PORST (output) is deasserted and HWCFG[3:5] pins are latched on PORST rising edge. Firmware execution is initiated.
  - T2 refers to the point in time when Firmware execution is completed. User code execution starts with a default frequency of 100 MHz.
  - T3 refers to the point in time during the Ramp-down phase when atleast one of the externally provided supplies (1.3 V or 3.3 V) drop below their respective primary under-voltage reset thresholds.

8.3.1.9 EVR Control Registers

All EVR registers with LCK bits have corresponding shadow registers in the EVR Standby domain. When writing consecutive EVR registers with LCK bits, it need to be ensured that the consecutive register is written only when the respective LCK bit is in unlocked state. This is because all EVR shadow registers are addressed using a shared bus across multiple clock domains and power isolation. An ongoing write will block the complete bus inhibiting parallel or consecutive writes on other registers until LCK bit is released. After a cold PORST, warm and system resets these registers may return a value 0 or the updated value by the Firmware. The reset value reflects the default isolation value in the shadow register. Otherwise, a read will provide the value of the most recent write operation. The status registers **EVRSTAT** and **EVRADCSTAT** are updated during Start-up and after every EVR measurement cycle and the reset value is dependent on the EVR topology. Therefore the value read from register may differ from the documented reset value.

EVRSTAT
EVR Status Register **($0B0_H$)** **Reset Value: $0000\ 1400_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		SCMOD		0	BGP ROK	0		UVS WD	0	UV1 3	OVS WD	0		OV1 3	EVR 13
r		rh		r	rh	r	r	rh	r	rh	rh	r		rh	rh

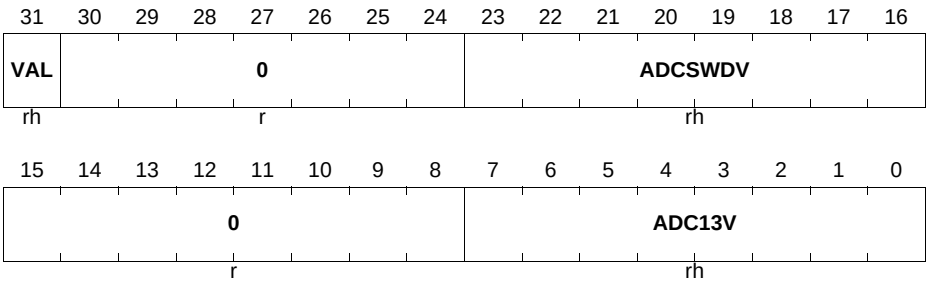
Field	Bits	Type	Description
EVR13	0	rh	**EVR13 status** This bit is set if the internal EVR13 LDO or SMPS regulator is active. EVR13 is activated if HWCFG[2] pin level is latched high during start-up phase. 0_B EVR13 is inactive. 1_B EVR13 is active.

Field	Bits	Type	Description
OV13	1	rh	**EVR13 Regulator Over-voltage event flag** This bit is set if EVR13 secondary voltage monitor recognises a 1.3 V over-voltage event. An alarm is raised to the SMU. 0_B No over-voltage condition happened. 1_B EVR13 Over-voltage event indication as configured in EVROVMON register.
OVSWD	4	rh	**Supply Watchdog (SWD) Over-voltage event flag** This bit is set if V_{DDP3} secondary voltage monitor recognises an over-voltage event. An alarm is raised to the SMU. 0_B No over-voltage condition happened. 1_B SWD Over-voltage event indication as configured in EVROVMON register.
UV13	5	rh	**EVR13 Regulator Under-voltage event flag** This bit is set if EVR13 secondary voltage monitor recognises a 1.3 V under-voltage event. An alarm is raised to the SMU. 0_B No under-voltage condition happened. 1_B EVR13 Under-voltage event indication as configured in EVRUVMON register.
UVSWD	7	rh	**Supply Watchdog (SWD) Under-voltage event flag** This bit is set if V_{DDP3} secondary voltage monitor recognises an under-voltage event. An alarm is raised to the SMU. 0_B No under-voltage condition happened. 1_B SWD Under-voltage event indication as configured in EVRUVMON register.
BGPROK	10	rh	**Primary Bandgap status** This bit is set after any reset and indicates that the primary bandgap voltage is in operational range. In case primary bandgap is not ok, we remain in cold reset state. On a Cold PORST release, this bit would be set to 1. 0_B Primary bandgap not ok. 1_B Primary bandgap is ok.

Field	Bits	Type	Description
SCMOD	[13:12]	rh	**Switch Capacitor SMPS Mode** If Switch Capacitor SMPS regulator is selected via HWCFG [0] pin, this bitfield indicates the conversion mode of the SC regulator. 00_B Mode 1/1 SC DCDC mode. 01_B Mode 1/2 SC DCDC mode. In case the current drive capabilities of the SC DCDC mode is exceeded, a switch from mode 2:1 to 1:1 takes place and an interrupt is issued.This bitfield shall be ignored incase EVR13 LDO mode is selected and is intended to be used only incase of EVR13 SCDCDC mode is selected.
0	[31:14], 11,9,6,3 ,2	r	**Reserved** Read as 0.

The over-voltage and under-voltage status signals are reported to SMU.EMM unit. An alarm for the upper and lower bound is supported in the SMU.EMM unit.

EVRADCSTAT
EVR ADC Status Register **($19C_H$)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
ADC13V	[7:0]	rh	**ADC 1.3 V Conversion Result** This bit field contains the last conversion result of the ADC measurement of the V_{DD} / EVR13 supply . VIN = [LSB * (ADC13V-1)] ; LSB = 5.77 mV Eg. 1.3 V – $E2_H$ – 226_D

Field	Bits	Type	Description
ADCSWDV	[23:16]	rh	**ADC External Supply Conversion Result** This bit field contains the last conversion result of the ADC measurement of the external V_{DDP3} supply. VIN = [LSB * (ADCSWDV-1)] ; LSB = 23.0 mV Eg. 3.3 V – 90_H – 144_D
VAL	31	rh	**Valid Status** This bit indicates if the register is being updated with a new value. Values are updated every EVR cycle. 0_B The register is not being updated 1_B The register is being updated
0	[30:24], [15:8]	r	**Reserved** Read as 0.

EVRRSTCON
EVR Reset Control Register ($06C_H$) Reset Value: 007E 00C5$_H$

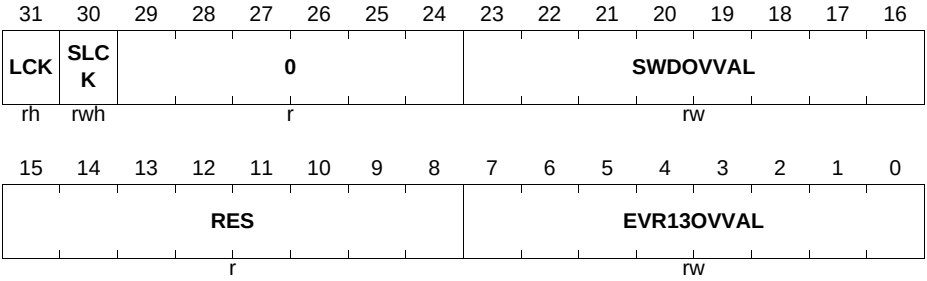
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LCK	**SLCK**	**BPRSTSWDOFF**	**RSTSWDOFF**	**RES**											
rh	rwh	w	rw	r											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES															
r															

Field	Bits	Type	Description
RSTSWDOFF	28	rw	**EVR SWD Reset Enable** 0_B A reset trigger signal is generated and forwarded to the SCU by the external supply monitoring block depending on the selected reset trim value. 1_B No reset trigger signal is generated and forwarded to the SCU by the external supply monitoring block depending on the selected reset trim value. This bit can only be changed if bit BPRSTSWDOFF is set in parallel.
BPRSTSWDOFF	29	w	**Bit Protection RSTSWDOFF** Setting this bit enables that bit RSTSWDOFF can be changed in this write operation. This bit is read as zero.
RES	[27:0]	r	**Reserved** These bits should not be written. Writes to other bitfields must ensure that these bits are not changed.
SLCK	30	rwh	**HSM Security Lock** 0_B No lock active 1_B Lock is active If this bit is set, all other bits in this register can no longer be written. Write requests to other bits when SLCK is set will cause a bus access error. SLCK bit can only be set by an access from the HSM master (TAG = 001101_B). A set operation performed by any other master or software is ignored and the bit is kept as cleared.
LCK	31	rh	**Lock Status** This bit indicates if the register can be updated with a new value or if the register is locked and a write action from the bus side has no effect. 0_B The register is unlocked and can be updated 1_B The register is locked and cannot be updated

EVROVMON

EVR Over-voltage Configuration Register(1A4$_H$) **Reset Value: 00F1 9EFD$_H$**



Field	Bits	Type	Description
EVR13OVVAL	[7:0]	rw	**1.3 V Regulator Over-voltage threshold** This field defines the over-voltage monitoring threshold level of the EVR13 regulator. 1.30 V – $E2_H$ – 226_D (refer datasheet) Threshold = [(VIN / LSB) + 1] LSB = 5.7692 mV
SWDOVVAL	[23:16]	rw	**Supply monitor (SWD) Over-voltage threshold value** This field defines the over-voltage threshold level of the external V_{DDP3} supply monitor. 3.3 V external supply 3.29 V – 90_H – 144_D (refer datasheet) Threshold = [(VIN / LSB) + 1] LSB = 23.077 mV The reset value of register is $F1_H$. Firmware later programs $9E_H$ to set the threshold to 3.3 V+10%.
SLCK	30	rwh	**HSM Security Lock** 0_B No lock active 1_B Lock is active If this bit is set, all other bits in this register can no longer be written. Write requests to other bits when SLCK is set will cause a bus access error. SLCK bit can only be set by an access from the HSM master (TAG = 001101_B). A set operation performed by any other master or software is ignored and the bit is kept as cleared.

Field	Bits	Type	Description
LCK	31	rh	**Lock Status** This bit indicates if the register can be updated with a new value or if the register is locked and a write action from the bus side has no effect. 0_B The register is unlocked and can be updated 1_B The register is locked and cannot be updated
0	[29:24],[15:8]	r	**Reserved** Read as 0; should be written with 0.

A configurable threshold with upper and lower voltage bounds can be defined in EVROVMON and EVRUVMON registers for monitoring external supply and EVR13 regulator output. It need to be considered that after a system reset, Firmware overwrites the register values.

The EVROVMON and EVRUVMON registers can be configured to trigger SMU alarms via software. It is also possible to activate alarms via software in the SMU unit.

EVRUVMON

EVR Under-voltage Configuration Register(1A0$_H$) **Reset Value: 00D0 89D6$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LCK	**SLCK**	**0**						**SWDUVVAL**							
rh	rwh	r						rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES								**EVR13UVVAL**							
r								rw							

Field	Bits	Type	Description
EVR13UVVAL	[7:0]	rw	**1.3 V Regulator Under-voltage threshold** This field defines the under-voltage monitoring threshold level of the EVR13 regulator. 1.30 V – E0$_H$ – 224$_D$ (refer datasheet) Threshold = [(VIN / LSB) - 1] LSB = 5.7692 mV The reset value of register is D6$_H$. Firmware programs CD$_H$.

Field	Bits	Type	Description
SWDUVVAL	[23:16]	rw	**Supply monitor (SWD) Under-voltage threshold value** This field defines the under-voltage threshold level of the external supply monitor. 3.3 V external supply 3.29 V – $8E_H$ – 142_D (refer datasheet) Threshold = [(VIN / LSB) + 1] LSB = 23.077 mV The reset value of register is $D0_H$. Firmware programs 84_H.
SLCK	30	rwh	**HSM Security Lock** 0_B No lock active 1_B Lock is active If this bit is set, all other bits in this register can no longer be written. Write requests to other bits when SLCK is set will cause a bus access error. SLCK bit can only be set by an access from the HSM master (TAG = 001101_B). A set operation performed by any other master or software is ignored and the bit is kept as cleared.
LCK	31	rh	**Lock Status** This bit indicates if the register can be updated with a new value or if the register is locked and a write action from the bus side has no effect. 0_B The register is unlocked and can be updated 1_B The register is locked and cannot be updated
0	[29:24],[15:8]	r	**Reserved** Read as 0; should be written with 0.

EVRMONCTRL
EVR Monitor Control Register (1A8$_H$) Reset Value: 0021 2121$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	SLCK	0						0		SWDUVMOD		0		SWDOVMOD	
r	rwh	r						r		rw		r		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								0		EVR13UVMOD		0		EVR13OVMOD	
r								r		rw		r		rw	

Field	Bits	Type	Description
EVR13OVMOD	[1:0]	rw	**1.3 V Regulator Over-voltage monitoring mode** 00_B Over-voltage monitoring does not trigger SMU alarm events but violation indicated in EVRSTAT register. 01_B An over-voltage event is triggered when the threshold is crossed in a lower to higher voltage transition 10_B An over-voltage event is triggered when the threshold is crossed in a higher to lower voltage transition 11_B An over-voltage event is triggered when the threshold is crossed in either direction
EVR13UVMOD	[5:4]	rw	**1.3 V Regulator Under-voltage monitoring mode** 00_B Under-voltage monitoring does not trigger SMU alarm events but violation indicated in EVRSTAT register. 01_B An under-voltage event is triggered when the threshold is crossed in a lower to higher voltage transition 10_B An under-voltage event is triggered when the threshold is crossed in a higher to lower voltage transition 11_B An under-voltage event is triggered when the threshold is crossed in either direction

Field	Bits	Type	Description
SWDOVMOD	[17:16]	rw	**Supply monitor (SWD) Over-voltage monitoring mode** 00_B Over-voltage monitoring does not trigger SMU alarm events but violation indicated in EVRSTAT register. 01_B An over-voltage event is triggered when the threshold is crossed in a lower to higher voltage transition 10_B An over-voltage event is triggered when the threshold is crossed in a higher to lower voltage transition 11_B An over-voltage event is triggered when the threshold is crossed in either direction
SWDUVMOD	[21:20]	rw	**Supply monitor (SWD) Under-voltage monitoring mode** 00_B Under-voltage monitoring does not trigger SMU alarm events but violation indicated in EVRSTAT register. 01_B An under-voltage event is triggered when the threshold is crossed in a lower to higher voltage transition 10_B An under-voltage event is triggered when the threshold is crossed in a higher to lower voltage transition 11_B An under-voltage event is triggered when the threshold is crossed in either direction
SLCK	30	rwh	**HSM Security Lock** 0_B No lock active 1_B Lock is active If this bit is set, all other bits in this register can no longer be written. Write requests to other bits when SLCK is set will cause a bus access error. SLCK bit can only be set by an access from the HSM master (TAG = 001101_B). A set operation performed by any other master or software is ignored and the bit is kept as cleared.

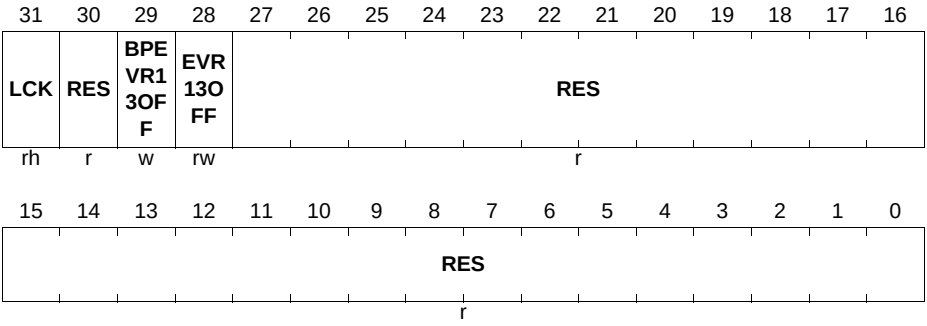
Field	Bits	Type	Description
0	2,3, [6:15], 18,19, [29:22], 31	r	**Reserved** Read as 0; should be written with 0.

The default setting after reset is that over-voltage indication is notified via an SMU alarm when the overvoltage threshold is crossed in a lower to higher voltage transition.

The default setting after reset is that under-voltage indication is notified via an SMU alarm when the under-voltage threshold is crossed in a higher to lower voltage transition.

It can be configured in EVRMONCTRL register to generate an interrupt when the over-under-voltage thresholds are crossed in either direction. This may be used to notify when the violation condition disappears with respect to secondary voltage monitoring.

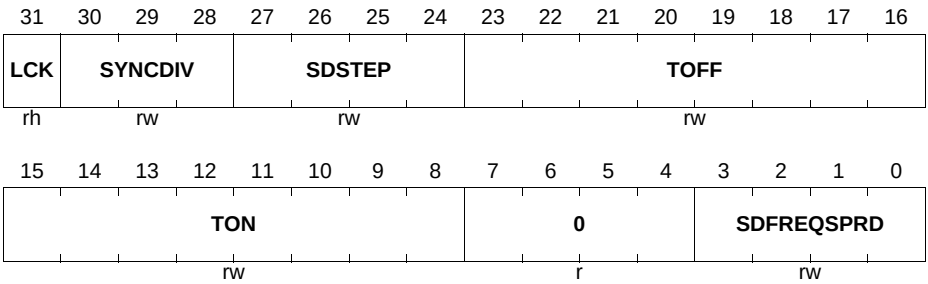
EVR13CON
EVR13 Control Register **($0B8_H$)** **Reset Value: 0380 0380_H**



Field	Bits	Type	Description
EVR13OFF	28	rw	**EVR13 Regulator Enable** 0_B The EVR13 regulator module is enabled 1_B The EVR13 regulator module is disabled/switched off. The device may be operating in standby or with external 1.3V supply. This bit can only be changed if bit BPEVR13OFF is set in parallel.

Field	Bits	Type	Description
BPEVR13OFF	29	w	**Bit Protection EVR13OFF** Setting this bit enables that bit EVR13OFF can be changed in this write operation. This bit is read as zero.
RES	30, [27:0]	r	**Reserved** These bits should not be written. Writes to other bitfields must ensure that these bits are not changed.
LCK	31	rh	**Lock Status** This bit indicates if the register can be updated with a new value or if the register is locked and a write action from the bus side has no effect. 0_B The register is unlocked and can be updated 1_B The register is locked and cannot be updated

EVRSDCTRL1
EVR13 SD Control Register 1 ($1B0_H$) Reset Value: $0829\ 2901_H$



Field	Bits	Type	Description
SDFREQSPRD	[3:0]	rw	**Frequency Spread Mode** This bit field activates frequency spreading for the SC SMPS regulator. The charge and discharge times are calculated every switching period as programmed in the TON and TOFF bitfields plus a uniform random offset. The maximum random offset clock cycles is defined by SDFREQSPRD field. 0_H No frequency spreading activated (default) 1_H 0 to 1 clock cycle is added in a random manner to both TON or TOFF switching periods (50% probability). 2_H 0 to 2 clock cycles are added. 3_H 0 to 4 clock cycles are added. 4_H 0 to 8 clock cycles are added. 5_H 0 to 16 clock cycles are added. All other values are reserved.
TON	[15:8]	rw	**Charge Phase length** The charge phase length in nominal 100 MHz clock cycles. Switching period = TON + TOFF + (16+2) cycles 1 MHz = 100 clock cycles 1.85 Mhz = 54 clock cycles
TOFF	[23:16]	rw	**Discharge Phase length** The discharge phase length in clock cycles. Switching period = TON + TOFF + (16+2) cycles 1 MHz = 100 clock cycles 1.85 Mhz = 54 clock cycles
SDSTEP	[27:24]	rw	**Droop Voltage Step** This bit field defines the voltage offset for droop compensation on a load jump.

Field	Bits	Type	Description
SYNCDIV	[30:28]	rw	**Clock Divider Ratio for external DCDC SYNC signal** This bitfield defines the divider factor for the clock signal to synchronise external regulator to the internal SC SMPS EVR13 regulator. 000_B $f_{DCDCSYNC} = f_{DCDC}$ 001_B $f_{DCDCSYNC} = f_{DCDC}/2$ 010_B $f_{DCDCSYNC} = f_{DCDC}/4$ 011_B $f_{DCDCSYNC} = f_{DCDC}/8$ 100_B $f_{DCDCSYNC} = f_{DCDC}/16$ 101_B $f_{DCDCSYNC} = f_{DCDC}/32$ All other combinations are reserved.
LCK	31	rh	**Lock Status** This bit indicates if the register can be updated with a new value or if the register is locked and a write action from the bus side has no effect. 0_B The register is unlocked and can be updated 1_B The register is locked and cannot be updated
0	[7:4]	r	**Reserved** Read as 0; should be written with 0.

EVRSDCTRL2
EVR13 SD Control Register 2 **($1B4_H$)** **Reset Value: $1541\ 9100_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LCK	**0**	**SDLUT**						**0**	**ADC LSB**	**ADCLPF**		**ADCMODE**			
rh	r	rw						r	rw	rw		rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIAD	**OL**	**NS**		**STSP**		**STBS**		**0**							
rw	rw	rw		rw		rw		r							

Field	Bits	Type	Description
STBS	[9:8]	rw	**Stabilization strength** The SMPS regulator is forced to regulate during zero crossing if this bit is set to avoid limit cycles.
STSP	[11:10]	rw	**Startup Speed** Selection of integrating speed at startup.
NS	[13:12]	rw	**Noise shaper setting** Selection of noise shaper for feedback ADC and whether the quantization error is considered. 00_B No noise shaping 01_B First order noise shaping 10_B Second order noise shaping 11_B Reserved
OL	14	rw	**Open Loop activation** Overwrite PI loop and operate in open loop.
PIAD	15	rw	**PI coefficient adaptation** Self adaptation of PI coefficients is activated.
ADCMODE	[19:16]	rw	**Operating Mode for ADC** Selection of the ADC mode. 01_B Tracking ADC mode All other bit combinations are reserved.
ADCLPF	[21:20]	rw	**Time constant of digital LPF of tracking ADC** Low pass filter for the measured ADC values. 00_B No filtering. 01_B Two values are averaged. 10_B Four values are averaged. 11_B Eight values are averaged.
ADCLSB	[22]	rw	**PID LSB size** Selection of resolution at the output of the low pass filter of the tracking ADC. 0_B 5 mV. 1_B 10 mV.
SDLUT	[29:24]	rw	**Non-linear Starting Point** Non linear slope setting.

Field	Bits	Type	Description
LCK	31	rh	**Lock Status** This bit indicates if the register can be updated with a new value or if the register is locked and a write action from the bus side has no effect. 0_B The register is unlocked and can be updated 1_B The register is locked and cannot be updated
0	30,23, [7:0]	r	**Reserved** Read as 0; should be written with 0.

EVRSDCTRL3
EVR13 SD Control Register 3 **($1B8_H$)** **Reset Value: $3C49\ 0880_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LCK	**MODHIGH**							**MODMAN**		**SDVOKLVL**					
rh	rw							rw		rw					

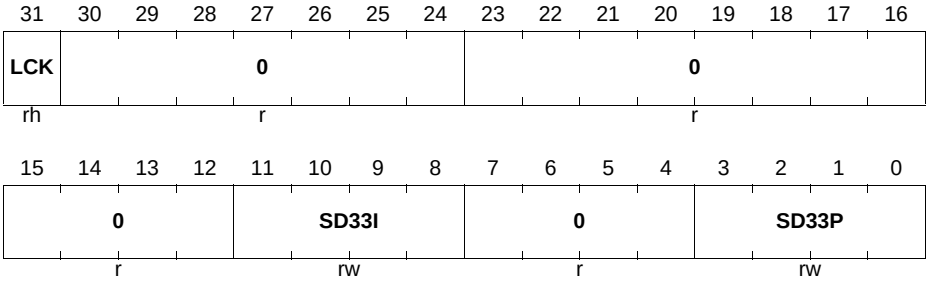
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**MODLOW**							**MOD SEL**	**SDOLCON**						
r	rw							rw	rw						

Field	Bits	Type	Description
SDOLCON	[6:0]	rw	**Initial Conductance** Value of Conductance for open loop operation. Activates parallel switches to increase conductance.
MODSEL	[7]	rw	**Operation Mode Selection** Manual or Automatic selection between modes. 0_B Automatic switching between modes enabled. 1_B Manual switching between modes enabled.
MODLOW	[14:8]	rw	**Low threshold for Mode change** The lowest value of conductance in closed loop operation before mode switch is triggered.
SDVOKLVL	[21:16]	rw	**Configuration of Voltage OK Signal** Filter and threshold configuration of the Voltage OK signal at startup.

Field	Bits	Type	Description
MODMAN	[23:22]	rw	**Manual Mode Selection** In case manual switching is selected in MODSEL bit, modes may be switched between the following 00_B Mode 1/1. 01_B Mode 1/2. All other combinations are reserved.
MODHIGH	[30:24]	rw	**High threshold for Mode change** The highest value of conductance in closed loop operation before mode switch is SMPS(1/2) to LDO(1/1) mode is triggered.
LCK	31	rh	**Lock Status** This bit indicates if the register can be updated with a new value or if the register is locked and a write action from the bus side has no effect. 0_B The register is unlocked and can be updated 1_B The register is locked and cannot be updated
0	15	r	**Reserved** Read as 0; should be written with 0.

EVRSDCOEFF2

EVR13 SD Coefficient Register 2 ($1C4_H$) Reset Value:0000 0507$_H$



Field	Bits	Type	Description
SD33P	[3:0]	rw	**P Coefficient** P control parameter for the PI regulator (3.3V external supply).

Field	Bits	Type	Description
SD33I	[11:8]	rw	**I Coefficient** I control parameter for the PI regulator (3.3V external supply).
LCK	31	rh	**Lock Status** This bit indicates if the register can be updated with a new value or if the register is locked and a write action from the bus side has no effect. 0_B The register is unlocked and can be updated 1_B The register is locked and cannot be updated
0	[30:12], [7:4]	r	**Reserved** Read as 0; should be written with 0.

8.3.2 Power Management

8.3.2.1 Power Management Overview

The Power Management scheme allows activation of power down modes so that the system operates with the minimum required power for the corresponding application state. A progressive reduction in power consumption is achieved by invoking Idle, Sleep or Standby modes respectively. The Idle mode is specific to each CPU where as Sleep and Standby modes influence the complete system.

As shown in **Table 8-20**, there are two power modes available for each CPU:

- CPU Run Mode
- CPU Idle Mode

Table 8-20 CPU Power Management

Mode	Description
Run Mode	The CPU clock is active and code is being executed.
Idle Mode	CPU may enter Idle Mode on following events: • On a SW Idle request issued by setting register bits **PMCSR0**.REQSLP = 01b when CPU has no active tasks to perform. • On a SMU Idle request in case CPU faults are detected triggering safety alarms which are configured in SMU to set CPU into Idle state. The CPU code execution is halted and CPU clock is disabled in Idle state. The peripherals continue to remain active.CPU RAM memories (PSPR / DSPR) are accessible to other bus masters and peripherals. CPU may exit Idle mode on following events: • When an interrupt occurs on a CPU returning the CPU to Run Mode. • When a trap occurs like an NMI trap event. • When the CPU watchdog or Safety watchdog timer overflow events trigger an SMU alarm in turn leading to a CPU interrupt. • When a MSB bit wrap of the CPU Watchdog counter takes place. • When an Application reset, System reset or any higher reset occurs.

As shown in **Table 8-21**, there are three main power modes available for the system:

- System Run Mode
- System Sleep Mode
- System Standby Mode

Table 8-21 System Power Management

Mode	Description
Run Mode	CPU has not requested Sleep Mode or Standby mode and is in Run mode. All peripheral modules are active.
Sleep Mode	System may enter Sleep Mode on following events: • On a SW Sleep request issued by setting **PMCSR0**.REQSLP = 10_B by the CPU. CPU code execution is halted and CPU Idle state is entered. Peripherals are set into sleep state if so configured in the respective CLCx.EDIS bit. Ports retain their earlier programmed state. System may exit Sleep mode on following events: • When an interrupt or trap is issued to a CPU. • When an NMI trap event takes place. • When the CPU watchdog or Safety watchdog timer overflow events trigger an SMU alarm leading in turn to a CPU interrupt. • When a MSB bit wrap of CPU Watchdog counter takes place. • When an Application reset, System reset or any higher reset occurs.
Standby Mode	System may enter Standby Mode on following events if so configured: • On a SW Standby request issued by setting **PMCSR0**.REQSLP= 11_B by the CPU. • On an NMI / $\overline{\text{ESR1}}$ assertion event. The Standby domain constituting the Standby RAM, the Wake-Up Timer, shared ports and the wake-up unit remain actively supplied. The power to the rest of the chip is completely switched off. System may exit Standby mode on following events in case V_{DDP3} remains supplied during Standby state: • when a wake-up edge is detected on selected pins / $\overline{\text{ESR1}}$. • when a wake-up request is issued by the Wake-Up Timer. • when $\overline{\text{PORST}}$ assertion is detected.

Furthermore, flexible reduction of power consumption is possible through following measures:

- Reduction of specific CPU power consumption by means of CPU throttling.
- Disabling the module clock by setting bit DISR in module CLC register.
- Reducing the system frequency without changing individual peripheral clocks.
- Reducing peripheral clock frequency without changing system clock frequency.

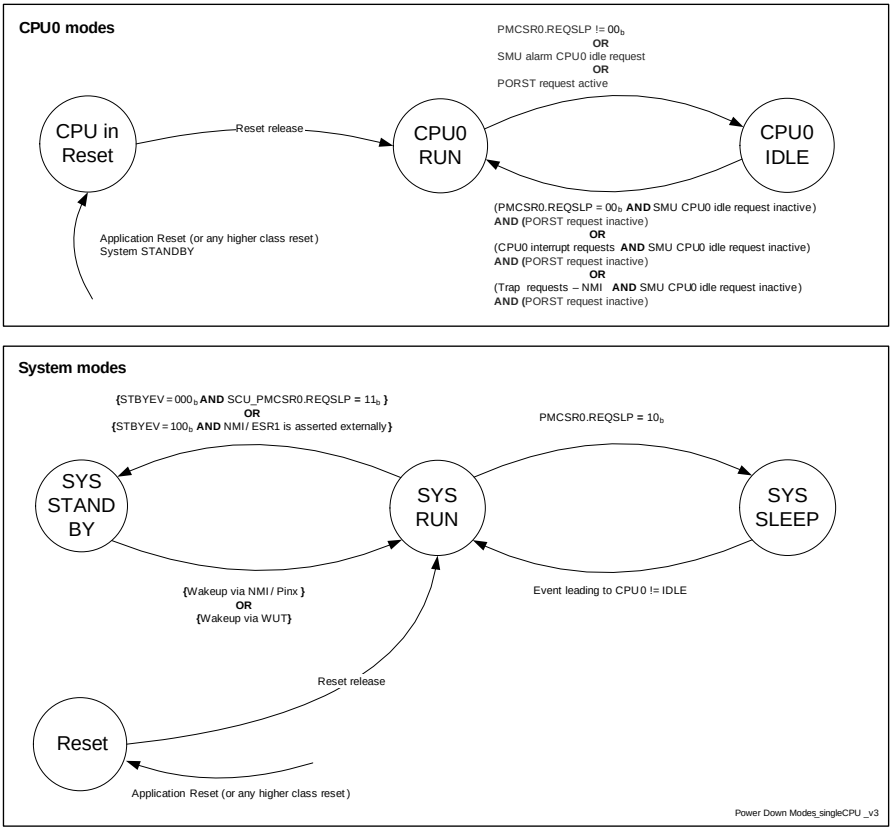


Figure 8-31 Power down modes and transitions

8.3.2.2 Idle Mode

In case there are no active tasks to perform, CPU may be requested to enter Idle mode during runtime by writing to the **PMCSR0** register and setting the bitfield REQSLP = 01_B.

Entering Idle Mode :

Following events can invoke a CPU Idle request

- CPUx setting itself in Idle by writing its own **PMCSR0** register: The **PMCSR0** register shall be accessed by setting CPUx ENDINIT = 0_B and consequently writing the bitfield REQSLP = 01_B. The Idle transition takes place only when CPUx ENDINIT = 1_B is set back again. This ensures that a CPUx does not enter Idle mode when it's WDTx is in Time-Out mode and ENDINITx = 0_B to avoid wake-up on a consequent WDT time-out. Safety ENDINIT mechanism shall not be used by a CPU to set itself into Idle to avoid wake-up on a Safety WDT time-out. It is to be noted that CPU is set into Idle state during entry into SLEEP (REQSLP = 10_B) mode as well. In STANDBY mode (REQSLP = 11_B), CPU is set first into Idle state before power is switched off.
- CPUx being set into Idle by SMU alarm: Idle mode may be requested by an SMU alarm in case of detected CPU faults thus setting the faulty CPU in Idle without initiating a full application reset.

The CPU watchdog may be disabled or slowed down by reprogramming the timers before triggering Idle request via Software. On an Idle request, the CPU finishes its current operations and sends an acknowledge back to the Power Management unit. It then enters an inactive state in which the CPU and the respective DMI and PMI memory units are shut off. The respective CPU clocks may be reduced via CPUxDIV register bitfield before issuing Idle request for further load jump reduction.

State during Idle mode

During Idle Mode, memory accesses to the DMI and PMI from other bus masters cause these units to wake-up automatically to handle these transactions. When memory transactions are complete, the DMI and PMI return to Idle state again. Once Idle Mode is entered, the state is reflected in **PMCSR0**.PMST status bits. CPU Idle request from the SMU is reflected in **PMCSR0**.SMUSLP bit.

Exiting Idle mode

In Idle mode, if there is no outstanding SMU Idle request to the CPU, the CPU will return to Run mode in response to the following events:

- An interrupt / trap received from an interrupt / trap source mapped to the CPU.
- An NMI trap request is received to wake-up the CPU.
- A MSB bit wrap of the corresponding CPU Watchdog counter occurs.
- Setting the register bits **PMCSR0**.REQSLP = 00_B to set the CPUx into Run mode.

If there is an outstanding SMU Idle request to the CPU, then the CPU may only return to Run mode after an application reset or any higher reset. The system enters reset state on an Application, System reset or any higher reset. If it is woken by a watchdog timer overflow event routed via the SMU to the CPU or by an NMI or by an interrupt, the CPU will immediately vector to the appropriate interrupt / trap handler.

8.3.2.3 Sleep Mode

System may be requested to enter Sleep mode by writing to the **PMCSR0** register and setting the bitfield REQSLP = 10_B.

The **PMCSR0** register shall be accessed by setting CPUx ENDINIT = 0_B. The Sleep request is issued only after CPUx ENDINIT bit is set back again. Safety ENDINIT mechanism shall not be used by a CPUx to issue Sleep request.

Entering Sleep Mode

It should be ensured to select individual clocks from Clock Control Unit for peripherals which need to remain active during Sleep mode. This allows the reduction of system clock frequencies , namely SRI and SPB clocks, to the minimum possible values via the low power divider. Low power modes for Analog and Flash modules may also be requested. The CLCx.EDIS register bit is cleared for all peripherals intended to be inactive in Sleep mode. The watchdogs may be disabled or slowed down before issuing a Sleep request.

State during Sleep Mode

Sleep Mode is disabled for a unit if CLCx.EDIS bit is set. The sleep request is ignored in this case and the corresponding unit continues normal operation as intended. If CLCx.EDIS is cleared, the clock of the module is gated. The exact sequence used for entering Sleep Mode is determined for each module by the setting of register field OCSx.SUS. CPU Idle state is entered as described in the previous section. All ports retain their earlier programmed state.

Exiting Sleep Mode

The system will exit Sleep mode on any wakeup event that causes the CPU to exit Idle Mode. The CPU would be set into Run mode (REQSLP = RUN, PMST = RUN). An NMI trap event will wake-up the system. A MSB bit wrap of the CPU Watchdog counter would also wake-up the CPU. The response of the CPU to being woken up from Sleep Mode is also the same as for Idle Mode. Peripheral units that have entered Sleep Mode will switch back to their selected Run Mode operation.

8.3.2.4 Standby Mode

The Standby domain constitutes the Standby RAM, the **W**ake**U**p **T**imer, the power management unit, the wake-up unit, the V_{DDP3} monitor and some basic infrastructure components. The Standby domain is supplied by the EVR pre-regulator and is by default clocked by the 100 KHz internal clock in Standby Mode.

Following events may trigger Standby mode entry based on **PMSWCR1**.STBYEV bits.

- Standby entry on SW request (**PMCSR0**.REQSLP = 11_B).
- Standby entry on ESR1 / NMI edge event.

Following events may trigger wake-up from Standby mode.

- Wake-up via NMI / Pinx: Wake-up on rising, falling or any edge of NMI / ESR1, Pin A or Pin B pins. Standby domain is supplied via V_{DDP3} supply pins which in turn is supplied by an external regulator with Standby support.
- Wake-up via WUT: Wake-up is triggered by the WakeUp Timer.

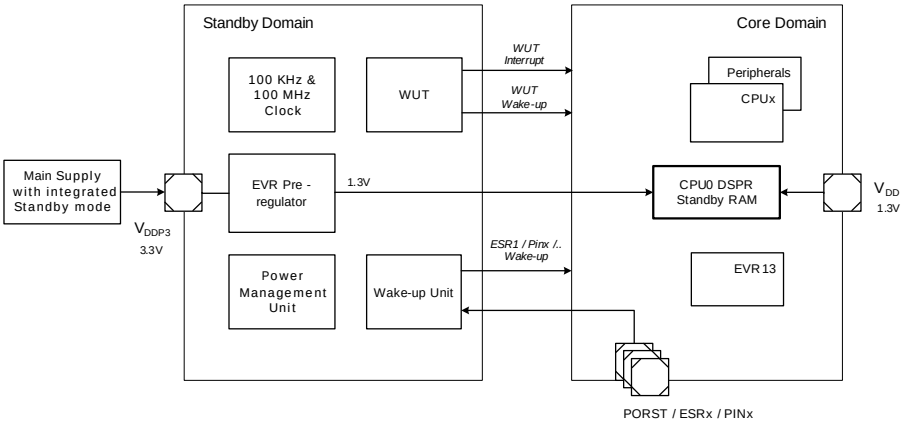


Figure 8-32 Standby domain

The Standby RAM constitutes a single block of ECC protected DSPR RAM of CPU0. The RAM remains supplied during Standby mode if configured in **PMSWCR0**.STBYRAMSEL bits. On wake-up, the status of the Standby RAM is reflected in **PMSWSTAT**.STBYRAM bits. On Standby request, the supply to the Standby RAM is immediately switched from EVR13 / V_{DD} supply to EVR Pre-regulator before Standby shut-off sequence is issued.

External events may be mapped to ESRx / PINx pins in turn acting as wake-up signals for the system. In Run Mode, NMI / ESR1 pin may be used as fault or functional interface for external devices. In Standby Mode, an edge event on the NMI / ESR1 pin may be configured to trigger wake-up of the main core domain via **PMSWCR0**.ESR1WKEN bit and is reflected in **PMSWSTAT**.ESR1WKEN status flag. It can be configured to trigger

a wake-up on rising, falling or both edges via **PMSWCR0**.ESR1EDCON bit. Glitches on ESR1 input are filtered out by activating the filter via **PMSWCR0**.ESR1DFEN bit. Additional pins (PINA - P14.1 and PINB - P15.1) may likewise be configured to trigger wake-up via **PMSWCR0**.xEDCON, xDFEN & xWKEN bits. On wake-up, **PMSWSTAT**.ESR1WKP or PINxWKP event flags provide information as to the wake-up source. It should be taken care after wake-up to clear the event flags via **PMSWSTATCLR** register. In case new wake-up events are captured while **PMSWSTAT**.xWKP flags are still set, then **PMSWSTAT**.xOVRRUN flags are set to indicate an overrun state owing to consecutive unserviced wake-up events.

Entering Standby Mode (ESR1 / Pin or WUT Wake-up configuration)

The Standby Mode entry may be requested by writing to **PMCSR0** register to set bitfield REQSLP = 11_B or via NMI / ESR1 assertion as configured in **PMSWCR1**.STBYEV bits.

Before entering standby mode, various modules should be sequentially shut off in a sequence mainly to avoid large current jump on standby entry.

- Peripheral module clocks are switched off in respective CLC.DISR registers. CPU watchdogs may be disabled or reconfigured for slower modes.
- CPU frequency reduction in steps compliant to load jump constraints.
- CPU code execution is switched from Flash to PSPR RAM. Flash modules are deactivated via FCON register.
- System Clock is switched to the internal 100 MHz clock source. System PLL & E-Ray PLL are switched off. Clock dividers are programmed to lower values.
- Clear **PMSWCR1**.IRADIS bit to disable Idle Request Acknowledge sequence activation for fast Standby Mode entry. This ensures that standby request is not blocked by a pending reset request / sequence.
- Select the Standby RAM block via **PMSWCR0**.STBYRAMSEL bits. Copy Standby RAM redundancy data (16 words in size) to DSPR0 starting at 0xD000 2000h. The procedure is also described in Boot ROM chapter in section “Preparation before to enter Standby mode”. Dcache write back to be executed before Standby entry.
- Select the clock source which need to be active on entry into Standby Mode via **PMSWCR0**.SCRCLKSEL bits.
- Configure the wake-up signals which should trigger exit from Standby Mode. Edge detection and filter activation is appropriately configured via **PMSWCR0**.xxxEDCON and **PMSWCR0**.xxxDFEN bits.
- Configure pad state via **PMSWCR0**.TRISTREQ bit. All pads may either be in tristate or have pull-up devices active. Configure **PMSWCR0**.ESR0TRIST bit to configure ESR0 behavior as reset output or tristate during Standby and on wake-up. In case HWCFG[2:5] pins are read by Firmware, it should be ensured to tie them to external pull devices.
- Enable ESR1 or PINx pins for wake-up via **PMSWCR0**.xxxWKEN bits.The wake-up from WUT should be activated via xxx bit. WUT should be running before entering the Standby state in case used.
- Configure Standby Entry event in **PMSWCR1**.STBYEV register bits.
- Standby request issued. An orderly shut down of various sub-systems is triggered to enter Standby mode.

State during Standby Mode

The Standby RAM (DSPR RAM of CPU0), the **W**ake**U**p **T**imer, the shared ports and the wake-up logic are kept alive in this mode. PORST pin serves as a wakeup source when asserted and shall be held high externally to remain in Standby state.

Exiting Standby Mode - Wake-up

The wake-up event may happen on

- ESR1 edge transition (NMI trap)
- Pin A or Pin B edge transition (P14.1 or P15.1)
- Wake-up on a WUT underflow

The main EVR13 regulators are ramped up on wake-up based on the configuration in **PMSWSTAT**.HWCFGEVR register bits. Firmware is executed thereafter including installation of RAM redundancy data. If Standby RAM was supplied during Standby state, it is indicated in **PMSWSTAT**.STBYRAM register bits. Additional RAM integrity checks may be carried out. RSTSTAT.STBYR bit indicates that the supply was reliable during Standby. The wake-up and over-run status flags are set in **PMSWSTAT** register and shall be cleared by **PMSWSTATCLR** register.

Exiting Standby mode - Power Fail or $\overline{\text{PORST}}$ assertion

The $\overline{\text{PORST}}$ pin triggers a wakeup from Standby if asserted low during Standby state. The device boots up ramping up the regulators followed by firmware execution similar to a normal device start-up.

The Standby RAM contents are kept intact after PORST assertion provided Firmware is not configured to re-initialize the Standby RAM depending on Flash configuration. Reset is propagated to external devices via the ESR0 pin on exit from Standby mode depending on **PMSWCR0**.ESR0TRIST configuration. Firmware may elongate ESR0 reset output depending on Flash configuration.

A power fail event of the Standby supply during Standby mode will inevitably result in the loss of Standby RAM contents. Consequently, a cold PORST event is issued and the Standby domain is set into reset. EVR Pre- regulator under-voltage violation is indicated in RSTSTAT.STBYR flag.

8.3.2.5 Wake-up Timer (WUT)

The Wake-up Timer is a basic low power counter which may be used to wake-up the system periodically from Standby mode. The timer may also be used during Run, Idle or Sleep modes. The following list enumerates the salient features.

- 24 bit counter running on 100 KHz clock source with programmable reload value.
- 24 bit counter status register providing the current count value.
- Timer resolution of 100 KHz or (100 KHz / 2^{10}) configured via a clock divider.
  - 10 us resolution : 10 us - 168 s range.
  - 10 ms resolution : 10 ms - 1.9 days range.
- 2 operating modes :
  - Auto Reload mode - WUT is started and stopped via Software. Automatic reload on counter underflow.
  - Standby Auto Stop mode - Counter starts counting down from reload value on Standby entry. Counter stops on underflow and triggers a system wake-up.
- Events on WUT counter underflow
  - Interrupt request on SRC_EVRWUT interrupt node on counter underflow during Run, Idle or Sleep mode.
  - Wake-up trigger on counter underflow during Standby mode.
  - Capture trigger on counter underflow to CCU6x_CC60IND for trimming purpose.
- Over-run indication of consecutive unserviced wake-up triggers.

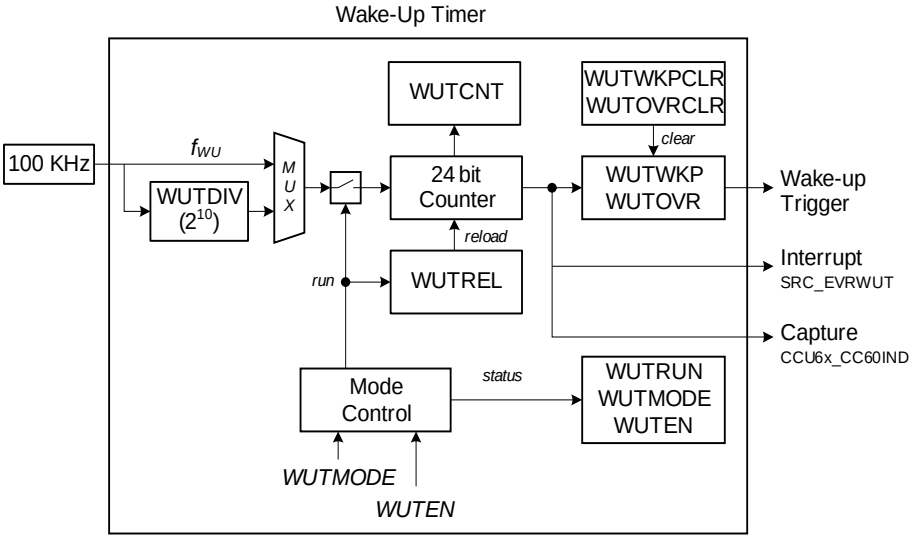


Figure 8-33 Wake-up Timer (WUT)

Wake-up Timer Operation

Wake-up Timer may be operated in following modes :

Table 8-22 Wake-up Timer Operation

WUT EN	WUT MODE	Mode Description
0B	XB	WUT is disabled and counter is stopped. **PMSWCR3**.WUTREL reload value may be updated. **PMSWSTAT**.WUTCNT,WUTRUN,WUTWKP & WUTOVR flags read 0.

Table 8-22 Wake-up Timer Operation

WUT EN	WUT MODE	Mode Description
1B	0B	Software Auto Reload mode : WUT starts running when **PMSWCR3**.WUTEN = 1 & WUTMODE = 0 is set. **PMSWSTAT**.WUTRUN bit is set indicating that WUT timer is currently running. **PMSWUTCNT**.WUTCNT bitfield indicates the actual counter value. On counter underflow, WUT is automatically reloaded with WUTREL value. During Standby, WUT underflow triggers system wake-up and **PMSWSTAT**.WUTWKP flag is set. During Run, Sleep or Standby modes, WUT underflow triggers an interrupt request and **PMSWSTAT**.WUTWKP flag is set. WUTREL reload value shall not be updated in this state. On wake-up, the **PMSWSTAT**.WUTWKP flag shall be cleared by **PMSWSTATCLR**.WUTWKPCLR bit.In case of unserviced consecutive counter underflow events, **PMSWSTAT**.WUTOVRUN flag is set to indicate an over-run wake-up event. Interrupt over-run event can be detected via SRC.IOV bit during RUN mode.
1B	1B	Standby Auto Stop mode: The mode is selected by setting **PMSWCR3**.WUTEN = 1 & WUTMODE = 1. WUT starts running only when Standby mode is entered.On counter underflow, WUT stops running and the wake-up of system is triggered and **PMSWSTAT**.WUTWKP flag is set. WUT starts running again on the next Standby mode entry. The intention is to have the timer running only during the Standby state. **PMSWUTCNT**.WUTCNT, **PMSWSTAT**.WUTRUN reads 0 after a wake-up.WUTREL reload value shall not be updated in this state. On wake-up, the **PMSWSTAT**.WUTWKP flag shall be cleared by **PMSWSTATCLR**.WUTWKPCLR bit. In case system was woken up by other wake-up triggers while WUT was still running, an interrupt request is generated on WUT underflow. In case of unserviced consecutive counter underflow events, **PMSWSTAT**.WUTOVRUN flag is set to indicate an over-run wake-up event. Interrupt over-run event can be detected via SRC.IOV bit during RUN mode.

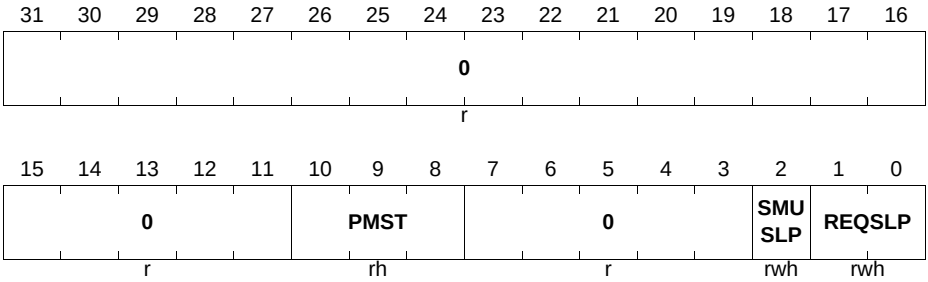
In case of timer overflow, an interrupt is issued on the interrupt node SRC_EVRWUT. Wake-up Timer reload value maybe trimmed during Run mode by comparing the time stamp on a WUT underflow captured by a GTM-TIM or CCU60 based on a more precise clock. This allows to compensate the 100 KHz (f_{WU}) clock source variations owing to technology and temperature.

8.3.2.6 Power Management Registers

Power Management Control and Status Register

The set of registers used for Power Management control the issue of power modes, manage wake-up configuration and provide status information on mode transitions. The request for Idle, Sleep or Standby mode is issued via **PMCSR0** register.

PMCSR0
Power Management Control and Status Register
($0D4_H$) **Reset Value: $0000\ 0100_H$**



Field	Bits	Type	Function
REQSLP	[1:0]	rwh	**Idle Mode and Sleep Mode Request** 00_B Request CPU Run Mode 01_B Request CPU Idle Mode 10_B Request System Sleep Mode 11_B Request System Standby Mode In Idle Mode or Sleep Mode, these bits are cleared in response to an interrupt for the CPU, or when bit 15 of the corresponding CPU Watchdog Timer register (bit WDTCPUxSR.TIM[15]) changes from 0 to 1. In Standby Mode, these bits are cleared on wake-up. REQSLP maybe written only when either CPU or Safety ENDINIT bits are reset to 0. Only after CPU or Safety ENDINIT bit is set back after REQSLP is written would the REQSLP take effect.

Field	Bits	Type	Function
SMUSLP	2	rwh	**SMU CPU Idle Request** 0_B No SMU Alarm request for CPU Idle Mode 1_B SMU Alarm requested CPU Idle Mode This bit is set to '1' when an SMU Alarm requests that the CPU is put into Idle Mode. This bit is NOT cleared in response to an interrupt to the CPU, or when bit 15 of the CPU Watchdog Timer register (bit WDTCPUxSR.TIM[15]) change from 0 to 1. It is cleared only by reset or by a write of '0' to this register. An attempt to write '1' has no effect.
0	3	r	**Reserved** Read as 0; should be written with 0.
PMST	[10:8]	rh	**Power management Status** This bit field reflects the current status of the CPU. 000_B Reserved, do not use this combination 001_B Normal Run Mode[1)] 010_B CPU Idle Mode requested 011_B CPU Idle Mode acknowledged 100_B Sleep Mode requested 101_B Reserved, do not use this combination 110_B Standby Mode requested 111_B Reserved, do not use this combination
0	[7:4], [31:11]	r	**Reserved** Read as 0; should be written with 0.

1) After a reset, all CPUs are in "Normal Run Mode", but this does not mean that all CPUs are executing code. This mode also includes the CPU "halt" mode which is the start-up default for all except CPU0.

Standby and Wake-up Control Registers

PMSWCR0

Standby and Wake-up Control Register 0 ($0C8_H$) **Reset Value: 0000 $02D0_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LCK	0	ESR0TRIST	0			DCDCSYNC	0	PORSTDF	0		WUTWKEN	0	STBYRAMSEL		0
rh	r	rw	r			rw	r	rw	r		rw	r	rw		r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PINBEDCON		PINBDFEN	PINAEDCON		PINADFEN	ESR1EDCON		ESR1DFEN	ESR0EDCON		ESR0DFEN	PINBWKEN	PINAWKEN	ESR1WKEN	0
rw		rw	rw		rw	rw		rw	rw		rw	rw	rw	rw	r

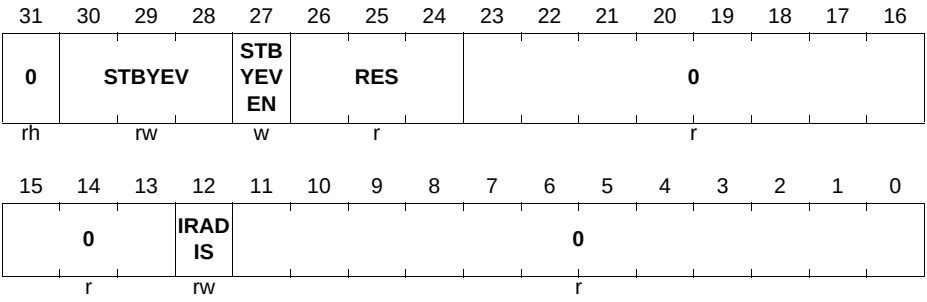
Field	Bits	Type	Function
0	0	r	**Reserved** Read as 0; should be written with 0.
ESR1WKEN	1	rw	**ESR1 Wake-up enable from Standby** 0_B System wake-up via ESR1 pin is disabled. 1_B System wake-up is enabled via ESR1 pin.
PINAWKEN	2	rw	**Pin A Wake-up enable from Standby** 0_B System wake-up via Pin A is disabled. 1_B System wake-up is enabled via Pin A.
PINBWKEN	3	rw	**Pin B Wake-up enable from Standby** 0_B System wake-up via Pin B is disabled. 1_B System wake-up is enabled via Pin B.
ESR0DFEN	4	rw	**Digital Filter Enable** This bit activates ESR0 digital spike filter. 0_B The filter is bypassed 1_B The filter is used

Field	Bits	Type	Function
ESR0EDCON	[6:5]	rw	**Edge Detection Control** This bit field defines the edge of a ESR0 wake-up trigger 00_B No trigger is generated 01_B A trigger is generated upon a rising edge 10_B A trigger is generated upon a falling edge 11_B A trigger is generated upon a rising OR falling edge
ESR1DFEN	7	rw	**Digital Filter Enable** This bit activates ESR1 digital spike filter. 0_B The filter is bypassed 1_B The filter is used
ESR1EDCON	[9:8]	rw	**Edge Detection Control** This bit field defines the edge of a ESR1 wake-up trigger 00_B No trigger is generated 01_B A trigger is generated upon a rising edge 10_B A trigger is generated upon a falling edge 11_B A trigger is generated upon a rising OR falling edge
PINADFEN	10	rw	**Digital Filter Enable** This bit activates Pin A digital spike filter. 0_B The filter is bypassed 1_B The filter is used
PINAEDCON	[12:11]	rw	**Edge Detection Control** This bit field defines the edge of a Pin A wake-up trigger 00_B No trigger is generated 01_B A trigger is generated upon a rising edge 10_B A trigger is generated upon a falling edge 11_B A trigger is generated upon a rising OR falling edge
PINBDFEN	13	rw	**Digital Filter Enable** This bit activates Pin B digital spike filter. 0_B The filter is bypassed 1_B The filter is used

Field	Bits	Type	Function
PINBEDCON	[15:14]	rw	**Edge Detection Control** This bit field defines the edge of a Pin B wake-up trigger 00_B No trigger is generated 01_B A trigger is generated upon a rising edge 10_B A trigger is generated upon a falling edge 11_B A trigger is generated upon a rising OR falling edge
0	16	r	**Reserved** Read as 0; should be written with 0.
STBYRAMSEL	[18:17]	rw	**Standby RAM supply in Standby Mode** 00_B Standby RAM is not supplied. 01_B Standby RAM (CPU0 DSPR) is supplied. *Note: All other bit combinations are reserved. Incase Standby RAM supply is activated, Standby RAM is not initialised after wakeup or PORST irrespective of configuration in PROCOND. RAMINSEL / RAMIN*
0	22,21, 19	r	**Reserved** Read as 0; should be written with 0.
WUTWKEN	20	rw	**WUT Wake-up enable from Standby** 0_B System wake-up via Wake-up Timer is disabled. 1_B System wake-up is enabled via Wake-up Timer.
PORSTDF	23	rw	**PORST Digital Filter enable** This bit field enables additional PORST digital filter to provide enhanced immunity against spurious spikes. 0_B PORST recognition delay = Analog PORST pad filter delay (default reset state). 1_B PORST recognition delay = Analog PORST pad filter delay + Digital filter delay.
0	24	r	**Reserved** Read as 0; should be written with 0.

Field	Bits	Type	Function
DCDCSYNC	25	rw	**DC-DC Synchronisation Enable** This bitfield enables the synchronisation output to synchronize the external SMPS regulator with respect to the internal SMPS EVR13 regulator. 0_B DC-DC Synchronisation signal not available. 1_B DC-DC Synchronisation signal available.
0	[28:26]	r	**Reserved** Read as 0; should be written with 0.
ESR0TRIST	29	rw	**ESR0 Tristate enable** This bit configures ESR0 pin behavior either as reset output or tristate during Standby mode. 0_B ESR0 configured as reset output and is held low during Standby state (default reset state) 1_B ESR0 in tristate during Standby state.
0	30	r	**Reserved** Read as 0; should be written with 0.
LCK	31	rh	**Lock Status** This bit indicates if the register can be updated with a new value or if the register is locked and a write action from the bus side has no effect. 0_B The register is unlocked and can be updated 1_B The register is locked and cannot be updated

PMSWCR1
Standby and Wake-up Control Register 1 ($0E8_H$) **Reset Value: $0100\ 0000_H$**

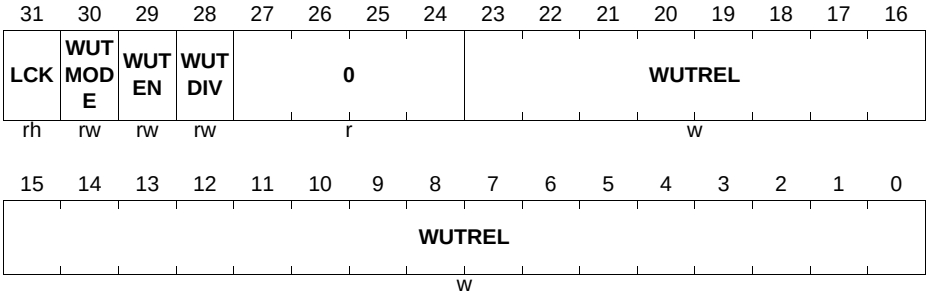


Field	Bits	Type	Description
0	[11:0]	r	**Reserved** Read as 0; should be written with 0.
IRADIS	12	rw	**Idle-Request-Acknowledge Sequence Disable** This bit enables SCU Idle Request Acknowledge sequence to all modules on Standby entry. This contributes an additional delay of 180us. 0_B Idle-Request-Acknowledge Sequence skipped on Standby entry. 1_B Idle-Request-Acknowledge Sequence issued on Standby entry. This bit shall be set before Standby entry to disable Idle request acknowledge sequence so that standby request is not blocked by a pending reset request / sequence.
0	[23:13]	r	**Reserved** Read as 0; should be written with 0.
RES	[26:24]	r	**Reserved** These bits shall not be written. When other bitfields are updated, it shall be ensured that these bits are not changed.
STBYEVEN	27	w	**Standby Entry Event configuration enable** 0_B Bit STBYEV is not updated. 1_B Bit STBYEV can be updated.
STBYEV	[30:28]	rw	**Standby Entry Event Configuration** 000_B Standby Entry triggered by setting **PMCSR0**.REQSLP register bit (Default). 100_B Standby Entry triggered on $\overline{\text{ESR1}}$/NMI assertion. *Note: All other bit combinations are reserved.*
0	31	r	**Reserved** Read as 0; should be written with 0.

Additional $\overline{\text{PORST}}$ digital filter activated via PMSWCR0.PORSTDF bit provides additional filtering of atleast 500 ns to provide enhanced immunity against spurious spikes. This is in addition to the inherent analog $\overline{\text{PORST}}$ filter delay of the PORST pad / pin as documented in the datasheet. After cold $\overline{\text{PORST}}$ this delay is by default inactive.

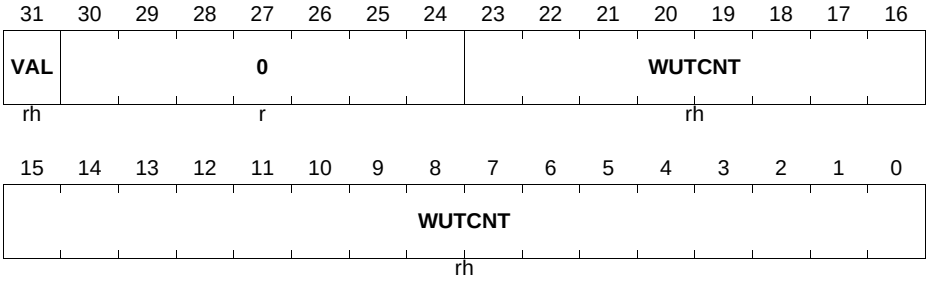
PMSWCR3

Standby and Wake-up Control Register 3(300_H) **Reset Value: $0000\ 0000_H$**



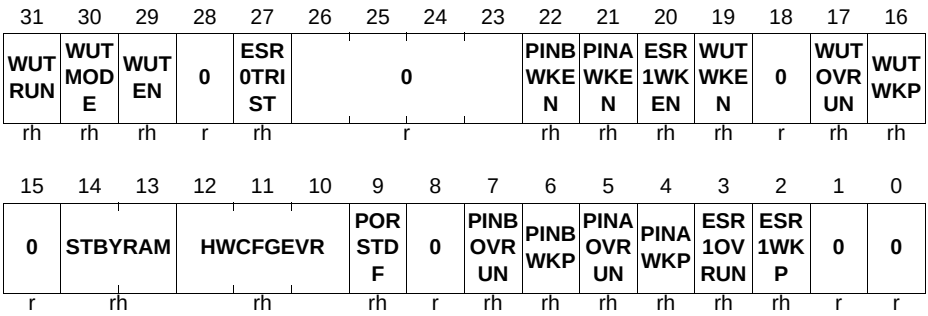
Field	Bits	Type	Description
WUTREL	[23:0]	rw	**WUT reload value.** The counter starts counting down from WUTREL value.
WUTDIV	28	rw	**WUT clock divider** 0_B WUT clock = f_{WU} = 100 KHz clock. 1_B WUT clock = f_{WU} (100 KHz) / 2^{10}.
WUTEN	29	rw	**WUT enable** This bit enables the Wake-up Timer. 0_B WUT is disabled. 1_B WUT is enabled.
WUTMODE	30	rw	**WUT mode selection** This bit selects the Wake-up Timer operation mode. 0_B WUT auto reload mode selected 1_B WUT auto stop mode selected.
LCK	31	rh	**Lock Status** This bit indicates if the register can be updated with a new value or if the register is locked and a write action from the bus side has no effect. 0_B The register is unlocked and can be updated 1_B The register is locked and cannot be updated
0	[27:24]	r	**Reserved** Read as 0; should be written with 0.

PMSWUTCNT
Standby WUT Counter Register (1DC$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
WUTCNT	[23:0]	rh	**WUT counter value.** The current WUT counter value is updated every 100 KHz.
VAL	31	rh	**Valid Status** This bit indicates if the register is being updated with a new value. Values are updated every EVR cycle. 0_B The register is not being updated 1_B The register is being updated
0	[30:24]	r	**Reserved** Read as 0; should be written with 0.

PMSWSTAT
Standby and Wake-up Status Flag Register(0CC$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
0	[1:0]	r	**Reserved** Read as 0; should be written with 0.
ESR1WKP	2	rh	**ESR1 Wake-up flag** 0_B No wake-up event detected on ESR1 input. 1_B An event as defined by PMSWCR0. ESR1EDCON detected on ESR1 input.
ESR1OVRUN	3	rh	**ESR1 Overrun status flag** This flag indicates that an ESR1 wake-up event occurred while the last one was not yet serviced. 0_B No overrun condition detected on ESR1 input. 1_B An overrun condition detected on ESR1 input.
PINAWKP	4	rh	**Pin A (P14.1) Wake-up flag** 0_B No wake-up event detected on Pin A input. 1_B An event as defined by PMSWCR0. PINAEDCON detected on Pin A input.
PINAOVRUN	5	rh	**Pin A Overrun status flag** This flag indicates that an PIN A wake-up event occurred while the last one was not yet serviced. 0_B No overrun condition detected on Pin A input. 1_B An overrun condition detected on Pin A input.
PINBWKP	6	rh	**Pin B (P15.1) Wake-up flag** 0_B No wake-up event occurred on the Pin B input. 1_B An event as defined by PMSWCR0. PINBEDCON detected on Pin B input.
PINBOVRUN	7	rh	**Pin B Overrun status flag** This flag indicates that an PIN B wake-up event occurred while the last one was not yet serviced. 0_B No overrun condition detected on Pin B input. 1_B An overrun condition detected on Pin B input.
0	8	r	**Reserved** Read as 0; should be written with 0.

Field	Bits	Type	Description
PORSTDF	9	rh	**PORST Digital Filter status** This bit field indicates whether additional PORST digital filter is activated. This bit is updated when PMSWCR0.PORSTDF is updated. 0_B PORST recognition delay = Analog PORST pad filter delay (default reset state). 1_B PORST recognition delay = Analog PORST pad filter delay + Digital filter delay.
HWCFGEVR	[12:10]	rh	**EVR Hardware Configuration** This bit field indicates the supply configuration latched by the EVR from HWCFG[2,0] during a cold startup. The latched configuration is used during STANDBY-RUN transition to reselect EVR mode. In TC22x HWCFGEVR[0] bit is set to 1 by default as SMPS mode is not available in TC22x unlike TC23x. .
STBYRAM	[14:13]	rh	**Standby RAM Supply status** This bit field indicates whether Standby RAM is supplied during Standby Mode and to infer status after a wake-up event. This bit is updated when PMSWCR0.STBYRAMSEL is updated. 00_B Standby RAM is not supplied. 01_B Standby RAM (CPU0 DMI) is supplied. $1x_B$ Reserved, do not use this combination.
WUTWKP	16	rh	**WUT Wake-up flag** 0_B No wake-up event detected due to WUT underflow. 1_B A wake-up event was detected due to WUT underflow.
WUTOVRUN	17	rh	**WUT Overrun status flag** This flag indicates that a WUT wake-up event occurred while the last one was not yet serviced. 0_B No overrun condition detected of WUT events. 1_B An overrun condition detected of WUT events.
0	[19:18], 15	r	**Reserved** Read as 0; should be written with 0.

Field	Bits	Type	Description
WUTWKEN	19	rh	**WUT Wake-up enable status** This bit indicates that WUT is enabled to trigger wake-up from Standby. This bit is updated when PMSWCR0.WUTWKEN bit is updated. Due to synchronization to 100 KHz WUT clock, setting of flag may take upto 30 us. 0_B Wake-up from Standby via WUT is disabled. 1_B Wake-up from Standby via WUT is enabled.
ESR1WKEN	20	rh	**ESR1 Wake-up enable status** This bit indicates that ESR1 is enabled to trigger wake-up from Standby. This bit is updated when PMSWCR0.ESR1WKEN bit is updated. 0_B Wake-up from Standby via ESR1 is disabled. 1_B Wake-up from Standby via ESR1 is enabled.
PINAWKEN	21	rh	**Pin A Wake-up enable status** This bit indicates that PINA is enabled to trigger wake-up from Standby. This bit is updated when PMSWCR0.PINAWKEN bit is updated. 0_B Wake-up from Standby via PINA is disabled. 1_B Wake-up from Standby via PINA is enabled.
PINBWKEN	22	rh	**Pin B Wake-up enable status** This bit indicates that PINB is enabled to trigger wake-up from Standby. This bit is updated when PMSWCR0.PINBWKEN bit is updated. 0_B Wake-up from Standby via PINB is disabled. 1_B Wake-up from Standby via PINB is enabled.
0	[26:23]	r	**Reserved** Read as 0; should be written with 0.
ESR0TRIST	27	rh	**ESR0 pin status during Standby** This bit indicates if ESR0 pin is configured as reset output or tristate during Standby mode & transitions.This bit is updated when PMSWCR0. ESR0TRIST is updated. 0_B ESR0 configured as reset output and is held low during Standby state (default reset state) 1_B ESR0 in tristate during Standby state.
0	28	r	**Reserved** Read as 0; should be written with 0.

Field	Bits	Type	Description
WUTEN	29	rh	**WUT Enable status** This bit indicates whether WUT is enabled. This bit is updated when PMSWCR3.WUTEN bit is updated. Due to synchronization to 100 KHz WUT clock, setting of flag may take upto 30 us. 0_B Wake-up timer is disabled. 1_B Wake-up timer is enabled.
WUTMODE	30	rh	**WUT Mode status** This bit indicates the current WUT mode. This bit is updated when PMSWCR3.WUTMODE bit is updated. Due to synchronization to 100 KHz WUT clock, setting of flag may take upto 30 us. 0_B WUT auto reload mode is selected 1_B WUT auto stop mode is selected.
WUTRUN	31	rh	**WUT Run status** This bit indicates whether WUT is currenly running. Due to synchronization to 100 KHz WUT clock, setting of flag may take upto 30 us. 0_B Wake-up timer is inactive. 1_B Wake-up timer is currently running.

PMSWSTATCLR

Standby and Wake-up Status Clear Register($0D0_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								0					0	WUTOVRUNCLR	WUTWKPCLR
r								r					r	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								PINBOVRUNCLR	PINBWKPCLR	PINAOVRUNCLR	PINAWKPCLR	ESR1OVRUNCLR	ESR1WKPCLR	0	0
r								w	w	w	w	w	w	r	r

Field	Bits	Type	Description
0	[1:0]	r	**Reserved** Read as 0; should be written with 0.
ESR1WKPCLR	2	w	**ESR1 Wake-up indication flag clear** 0_B No action 1_B PMSWSTAT.ESR1WKP bit cleared.
ESR1OVRUNCLR	3	w	**ESR1 Overrun status indication flag clear** 0_B No action 1_B PMSWSTAT.ESR1OVRUN bit cleared.
PINAWKPCLR	4	w	**PINA Wake-up indication flag clear** 0_B No action 1_B PMSWSTAT.PINAWKP bit cleared.
PINAOVRUNCLR	5	w	**PINA Overrun status indication flag clear** 0_B No action 1_B PMSWSTAT.PINAOVRUN bit cleared.
PINBWKPCLR	6	w	**PINB Wake-up indication flag clear** 0_B No action 1_B PMSWSTAT.PINBWKP bit cleared.
PINBOVRUNCLR	7	w	**PINB Overrun status indication flag clear** 0_B No action 1_B PMSWSTAT.PINBOVRUN bit cleared.
WUTWKPCLR	16	w	**WUT Wake-up indication flag clear** 0_B No action 1_B PMSWSTAT.WUTWKP bit cleared. Due to synchronization to 100 KHz WUT clock, clearing of PMSWSTAT.WUTWKP flag may take upto 30 us. The SW has to check PMSWSTAT. WUTWKP bit to confirm whether the bit was cleared after WUTWKPCLR request was issued.
WUTOVRUNCLR	17	w	**WUT Overrun status indication flag clear** 0_B No action 1_B PMSWSTAT.WUTOVRUN bit cleared. Due to synchronization to 100 KHz WUT clock, clearing of PMSWSTAT.WUTOVRUN flag may take upto 30 us. The SW has to check PMSWSTAT. WUTOVRUN bit to check whether the bit was cleared after WUTOVRUNCLR request was issued.

Field	Bits	Type	Description
0	18, [15:8]	r	**Reserved** Read as 0; should be written with 0.
0	[23:19]	r	**Reserved** Read as 0; should be written with 0.
0	[31:24]	r	**Reserved** Read as 0; should be written with 0.

8.3.3 Power Management Register Address

Table 8-23 Registers Address Spaces - PMC Kernel Registers

Module	Base Address	End Address	Note
SCU	$F003\ 6000_H$	$F003\ 63FF_H$	-

8.3.4 PMC Kernel Registers

This section describes the kernel registers of the PMC module. Most of PMC kernel register names described in this section will be referenced in other parts of the TC21x/TC22x/TC23x User's Manual by the module name prefix "SCU_".

PMC Kernel Register Overview

Table 8-24 Overview of PMC Registers (Offset from SCU Register Base)

Short Name	Long Name	Offset Addr. [1]	Access Mode		Reset	Description See
			Read	Write		
EVRRSTCON	EVR Reset Control Register	$06C_H$	U, SV	SV, SE, P	Cold Power-on Reset[2]	Page 8-145
EVRSTAT	EVR Status Register	$0B0_H$	U, SV	BE	Cold Power-on Reset	Page 8-142
–	Reserved	$0B4_H$[3]	–	BE	–	–
EVR13CON	EVR13 Control Register	$0B8_H$	U, SV	SV, SE, P	Cold Power-on Reset	Page 8-152
–	Reserved	$0BC_H$[3]	–	nE	–	–
PMSWCR0	Standby and Wakeup Control Register 0	$0C8_H$	U, SV	SV, SE, P	Cold Power-on Reset	Page 8-175
PMSWSTAT	Standby and Wakeup Status	$0CC_H$	U,SV	BE	Cold Power-on Reset	Page 8-181
PMSWSTATCLR	Standby and Wakeup Status Clear	$0D0_H$	U,SV	SV, SE, P	Cold Power-on Reset	Page 8-185

Table 8-24 Overview of PMC Registers (Offset from SCU Register Base)

Short Name	Long Name	Offset Addr. [1]	Access Mode		Reset	Description See
			Read	Write		
PMCSR0	CPU0 Power Management Control and Status Register	$0D4_H$	U, SV	SE,CE0, SV, P	Application Reset	Page 8-173
–	Reserved	$0D8_H$ [3]	–	nE	–	–
–	Reserved	$0DC_H$ [3]	–	nE	–	–
PMSWCR1	Standby and Wakeup Control Register 1	$0E8_H$	U, SV	SV, SE, P	Cold Power-on Reset	Page 8-178
–	Reserved	$0EC_H$ [3]	–	nE	–	–
EVRADCSTAT	EVR ADC Status Register	$19C_H$	U, SV	BE	Power-on Reset	Page 8-144
–	Reserved	198_H	–	nE	–	–
EVRUVMON	EVR Undervoltage Monitor	$1A0_H$	U, SV	SV, SE, P	Cold Power-on Reset	Page 8-148
EVROVMON	EVR Overvoltage Monitor	$1A4_H$	U, SV	SV, SE, P	Cold Power-on Reset	Page 8-147
EVRMONCTRL	EVR Monitor Control Register	$1A8_H$	U, SV	SV, SE, P	Cold Power-on Reset	Page 8-150
–	Reserved	$1AC_H$	–	nE	–	–
EVRSDCTRL1	EVR SD Control Register 1	$1B0_H$	U, SV	SV, SE, P	Cold Power-on Reset	Page 8-153
EVRSDCTRL2	EVR SD Control Register 2	$1B4_H$	U, SV	SV, SE, P	Cold Power-on Reset	Page 8-155
EVRSDCTRL3	EVR SD Control Register 3	$1B8_H$	U, SV	SV, SE ,P	Cold Power-on Reset	Page 8-157
–	Reserved	$1BC_H$	–	nE	–	–

Table 8-24 Overview of PMC Registers (Offset from SCU Register Base)

Short Name	Long Name	Offset Addr. 1)	Access Mode		Reset	Description See
			Read	Write		
–	Reserved	$1C0_H$ 3)	–	nE	–	–
EVRSDCOEFF2	EVR SD Coefficient Register 2	$1C4_H$	U, SV	SV, SE, P	Cold Power-on Reset	Page 8-158
–	Reserved	$1C8_H$-$1D4_H$ 3)	–	nE	–	–
PMSWUTCNT	Standby WUT Count Register	$1DC_H$	U, SV	BE	Cold Power-on Reset	Page 8-181
PMSWCR3	Standby and Wakeup Control Register 3	300_H	U, SV	SV, SE, P	Cold Power-on Reset	Page 8-180
–	Reserved	304_H - $37C_H$	BE	BE	–	–

1) The absolute register address is calculated as follows:
Module Base Address + Offset Address (shown in this column)

2) All EVR registers excluding status registers have corresponding shadow registers in the EVR Standby domain. After a cold PORST, a read of these registers may return the reset value or the value updated by the Firmware. The reset value reflects the default isolation value in the shadow register. Otherwise, a read will provide the value of the most recent write operation.

3) The address does not generate a bus error on read/write access.

8.4 System Control Unit (SCU)

The System Control Unit (SCU) of the TC21x/TC22x/TC23x contains miscellaneous control registers associated with other system functions.

This includes:

- External Request and Cross-triggering Unit (ERU) (see **Section 8.4.1**)
- CPU Lockstep Comparator Logic (LCL) (see **Section 8.4.2**)
- Die Temperature Sensor (DTS) (see **Section 8.4.3**)
- Watchdog Timers (WDTx) (see **Section 8.4.4**)
- Emergency Stop (EMS) (see **Section 8.4.5**)
- Logic Built-in-Self-Test (LBIST) (see **Section 8.4.6**)
- Overlay Control (OVC) (see **Section 8.4.7**)
- Miscellaneous System Control Registers (see **Section 8.4.8**)
- SCU register overview table (see **Table 8-29**)

8.4.1 External Request Unit (ERU)

The External Request Unit (ERU) is a versatile event and pattern detection unit. Its major task is the **generation of interrupts based on selectable trigger events at different inputs**, e.g. to generate external interrupt requests if an edge occurs at an input pin. The detected events can also be used by other modules to trigger or to gate module-specific actions.

8.4.1.1 Introduction

The ERU of the TC21x/TC22x/TC23x can be split in three main functional parts:

- 8 independent **Input Channels x** for input selection and conditioning of trigger or gating functions
- Event distribution: A **Connecting Matrix** defines the events of the Input Channel x that lead to a reaction of an Output Channel y.
- 8 independent **Output Channels y** for combination of events, definition of their effects and distribution to the system (interrupt generation, ...)

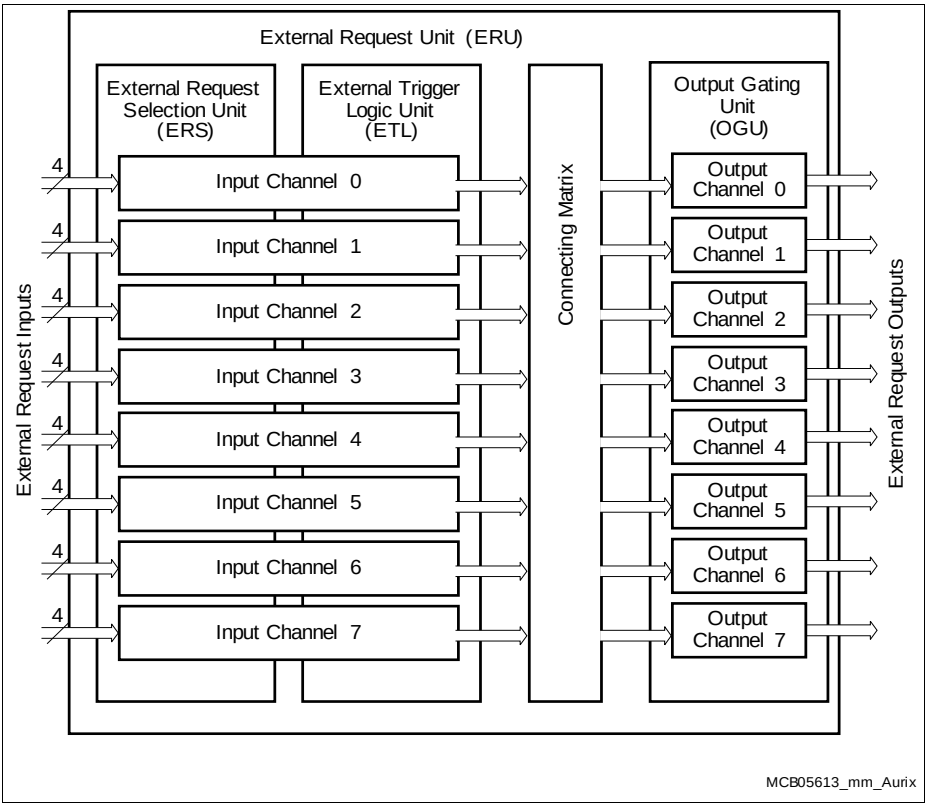


Figure 8-34 External Request Unit Overview

These tasks are handled by the following building blocks:

- An **External Request Select Unit (ERSx)** per Input Channel allows the selection of one input vector out of the 4 possible available inputs.
- An **Event Trigger Logic (ETLx)** per Input Channel allows the definition of the transition (edge selection, or by software) that lead to a trigger event and can also store this status. Here, the input levels of the selected signals are translated into events (event detected = event flag becomes set, independent of the polarity of the original input signals).
- The **Connecting Matrix** distributes the events and status flags generated by the Input Channels to the Output Channels.
- An Output Gating Unit (OGUy) per Output Channel that combines the available trigger events and status information from the Input Channels. An event of one Input Channel can lead to reactions of several Output Channels, or also events of several

Input Channels can be combined to a reaction of one Output Channel (pattern detection).
Different types of reactions are possible, e.g. interrupt generation (based on signals ERU_IOUTy).

8.4.1.2 ERU Input Pin Connections

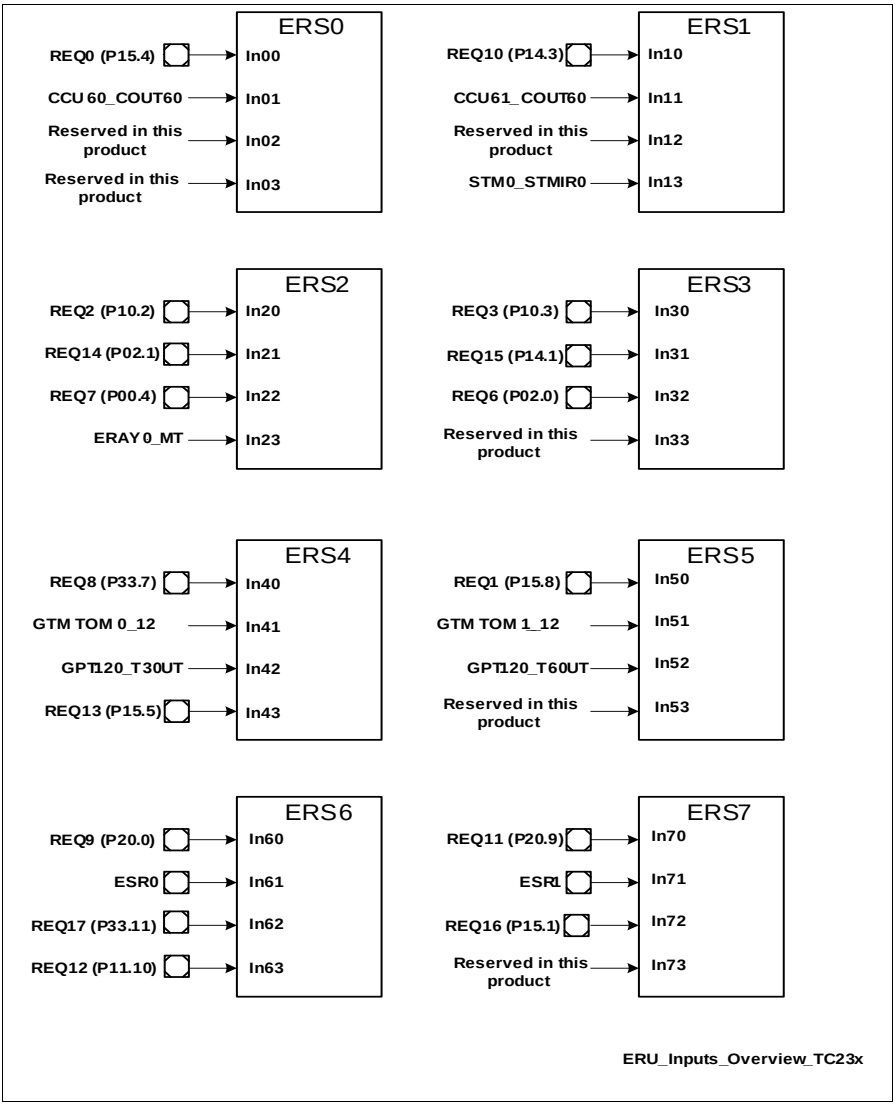


Figure 8-35 External Request Unit Input Connections for TC22x/TC23x

The inputs to the ERU can be selected from a large number of input signals. 16 of these inputs come directly from input REQx ports, but other inputs come from various peripheral module status signals. Usually, such inputs would be selected for an ERU function when the module input function is not used by the application, or when the module is not used at all. However, it is also possible to select an input which is also used by the other module, to also be used in the ERU as a trigger or to be combined with other signals (e.g. to generate an interrupt trigger when a start-of-frame is detected on a selected communication interface input).

8.4.1.3 External Request Selecter Unit (ERS)

Each ERS selects one of four inputs as the one input signal of the respective input channel. **Figure 8-36** shows the structure of this block.

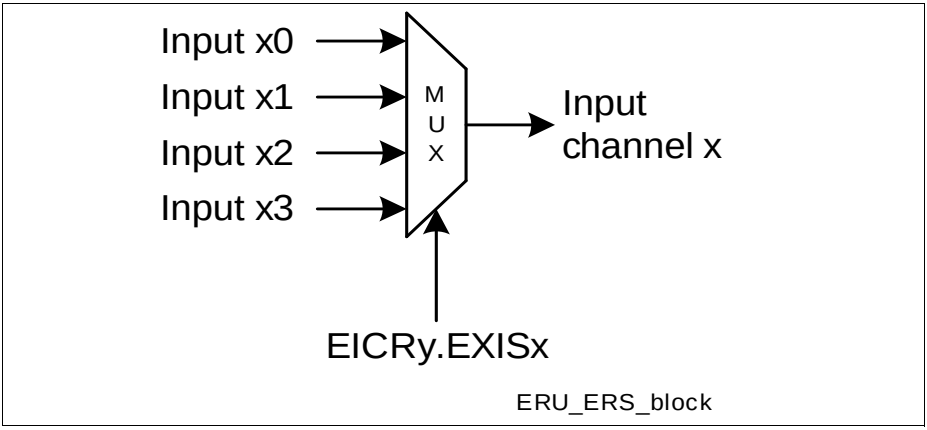


Figure 8-36 External Request Select Unit Overview

The ERS unit for channel x is controlled via bit field EICRy.EXISx.

8.4.1.4 Event Trigger Logic (ETL)

For each Input Channel x, an event trigger logic ETLx derives a trigger event and a status from the input channel x delivered by the associated ERSx unit. Each ETLx is based on an edge detection block, where the detection of a rising or a falling edge can be individually enabled. Both edges lead to a trigger event if both enable bits are set (e.g. to handle a toggling input).
Each pair of the four ETL units has an associated EICRy register, that controls all options of an ETL (the register also holds control bits for the associated ERS unit pair, e.g. EICR0 to control ESR0 and ESR1 plus and ETL0 and ETL1).

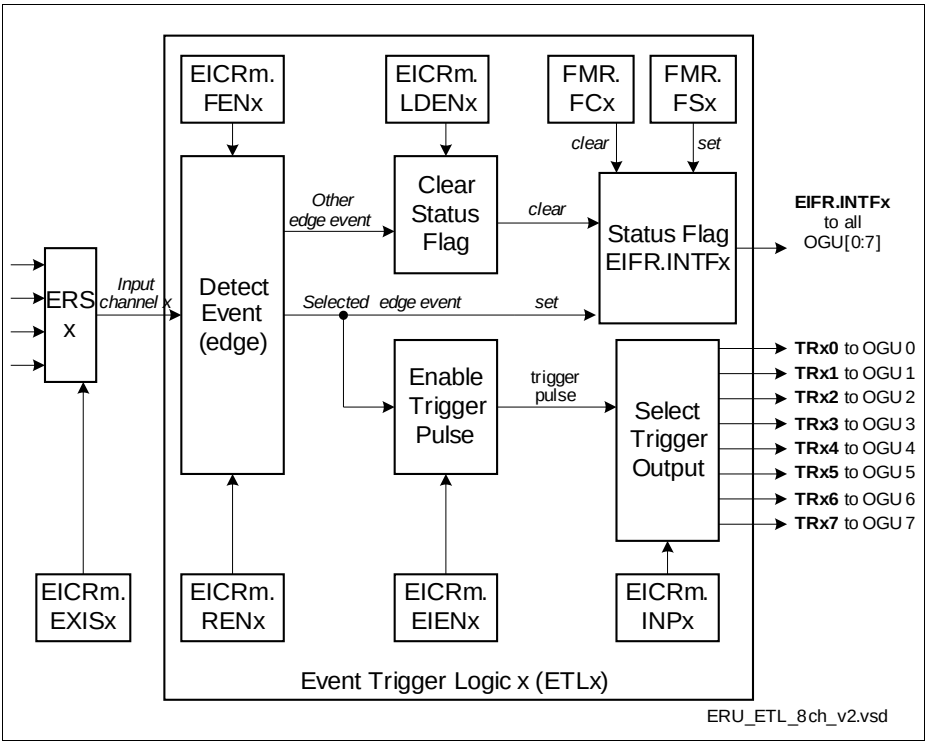


Figure 8-37 Event Trigger Logic Overview

When the selected event (edge) is detected, the status flag EIFR.INTFx becomes set. The status flag is cleared automatically if the “opposite” event is detected, if so enabled via bit EICRy.LDENx = 1. For example, if only the falling edge detection is enabled to set the status flag, it is cleared when the rising edge is detected. In this mode, it can be used for pattern detection where the actual status of the input is important (enabling both edge detections is not useful in this mode).
The output of the status flag is connected to all following Output Gating Units (OGUz) in parallel (see **Figure 8-38**) to provide **pattern detection capability of all OGUz** units based on different or the same status flags.

In addition to the modification of the status flag, a trigger pulse output TRxz of ETLx can be enabled (by bit EICRy.EIENx) and selected to **trigger actions in one of the OGUz** units. The target OGUz for the trigger is selected by bit field EICRy.INPx.
The trigger becomes active when the selected edge event is detected, independently from the status flag EIFR.INTFx.

8.4.1.5 Connecting Matrix

The connecting matrix distributes the trigger signals (TRxy) and status signals (EIFR.INPFx) from the different ETLx units between the OGUy units. **Figure 8-38** provides a complete overview of the connections between the ETLx and the OGUz units.

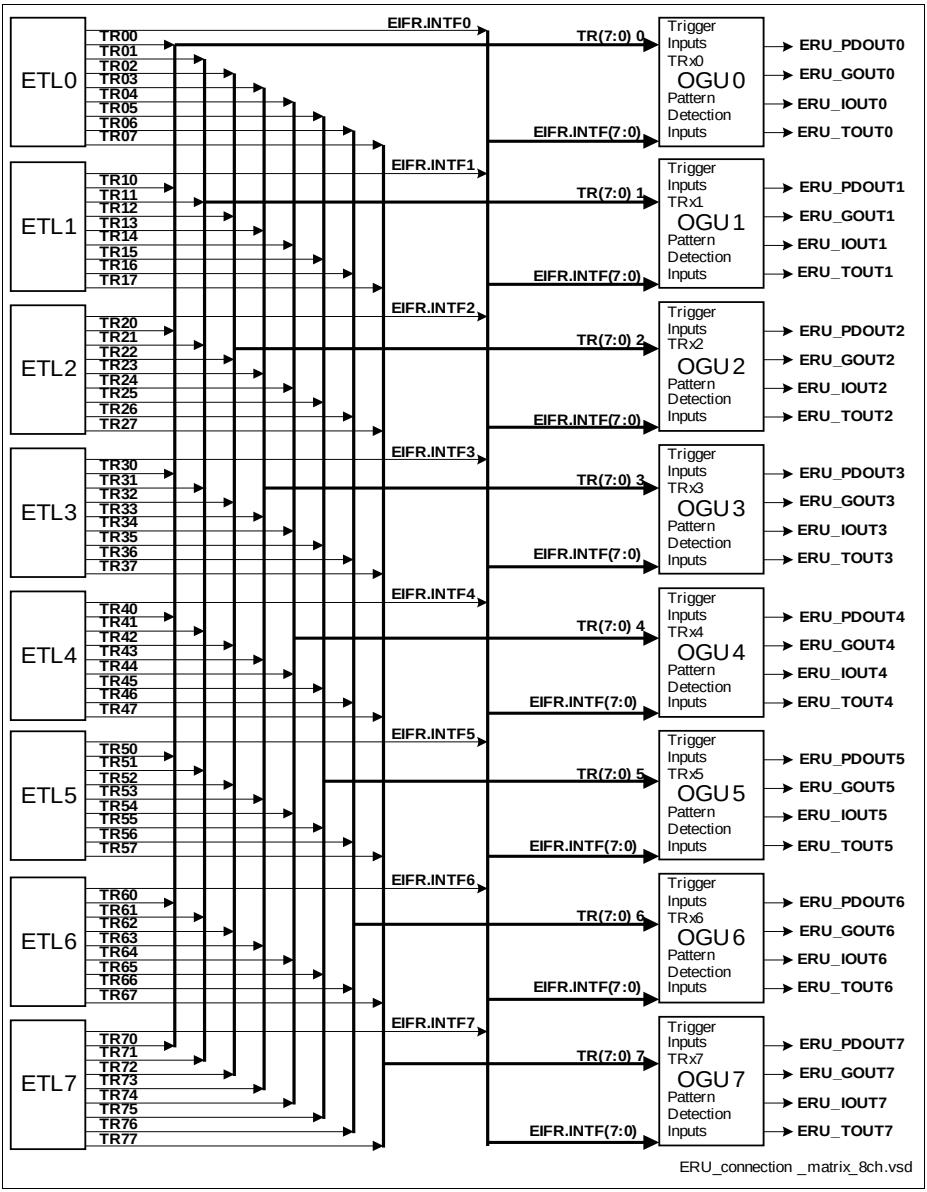


Figure 8-38 Connecting Matrix between ETLx and OGUy

8.4.1.6 Output Gating Unit (OGU)

Each OGUy unit combines the available trigger events and status flags from the Input Channels and distributes the results to the system. **Figure 8-39** illustrates the logic blocks within an OGUy unit. All functions of an OGUy unit are controlled by the associated IGCRm registers, one for each pair of output channels e.g. IGCR1 for OGU2 and OGU3. The function of an OGUy unit can be split into two parts:

- **Trigger combination**:
  All trigger signals TRxy from the Input Channels that are enabled and directed to OGUy and a pattern change event (if enabled) are logically OR-combined.
- **Pattern detection**:
  The status flags EIFR.INTFx of the Input Channels can be enabled to take part in the pattern detection. A pattern match is detected while all enabled status flags are set.

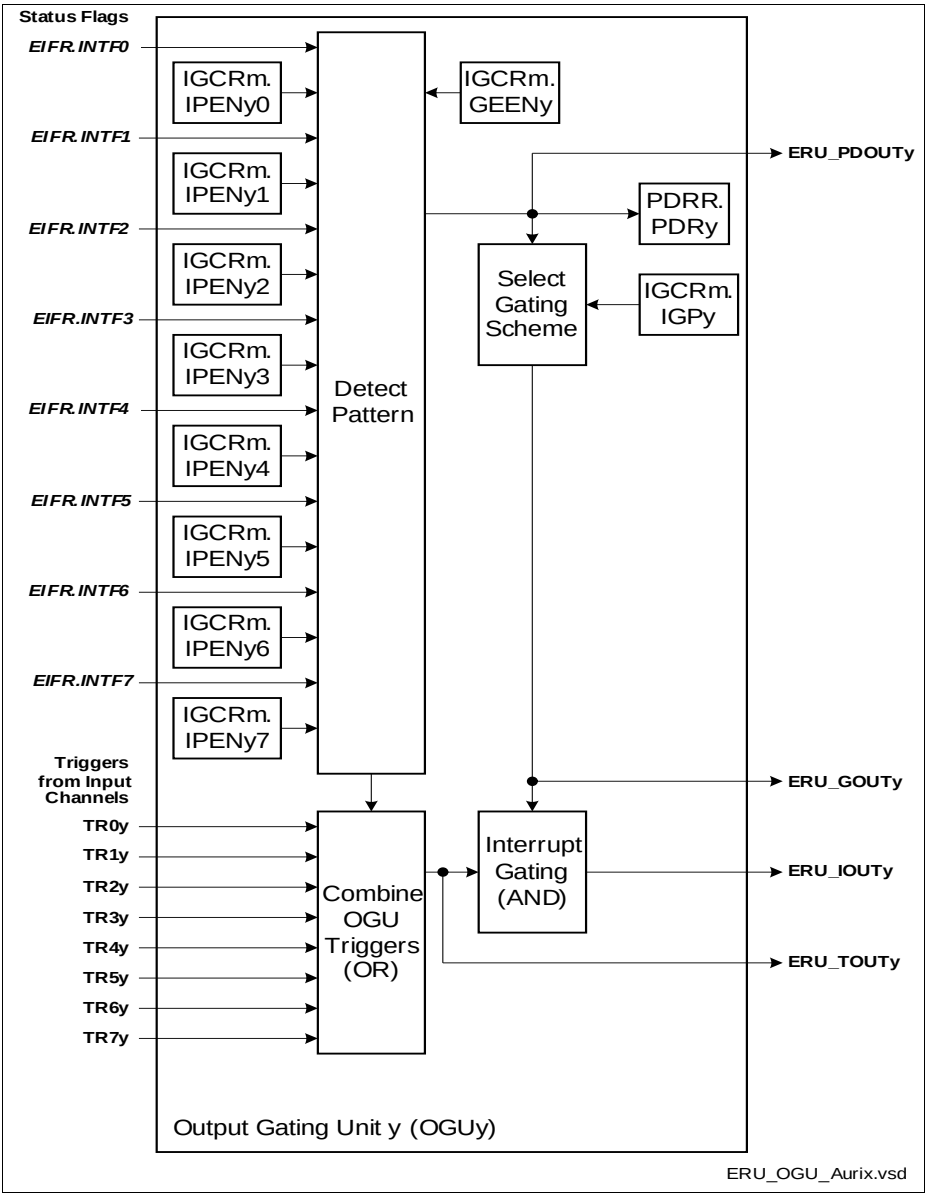


Figure 8-39 Output Gating Unit for Output Channel y

Each OGUy units generates 4 output signals that are distributed to the system.

- **ERU_PDOUTy** to directly output the pattern match information for gating purposes in other modules (pattern match = 1).
- **ERU_GOUTy** to output the pattern match or pattern miss information (inverted pattern match), or a permanent 0 or 1 under software control for gating purposes in other modules.
- **ERU_TOUTy** as combination of a peripheral trigger, a pattern detection result change event, or the ETLx trigger outputs TRxy to trigger actions in other modules.
- **ERU_IOUTy** as gated trigger output (ERU_GOUTy logical AND-combined with ERU_TOUTy) to trigger interrupts (e.g. the interrupt generation can be gated to allow interrupt activation during a certain time window).

Trigger Combination

The trigger combination logically OR-combines different trigger inputs to form a common trigger ERU_TOUTy. Possible trigger inputs are:

- In each ETLx unit of the **Input Channels**, the trigger output TRxy can be enabled and the trigger event can be directed to one of the OGUy units.
- In the case that at least one **pattern detection** input is enabled (IGCRm.IPENxy) and a change of the pattern detection result from pattern match to pattern miss (or vice-versa) is detected, a trigger event is generated to indicate a pattern detection result event (if enabled by IGCRm.GEENy).

The trigger combination offers the possibility to program different trigger criteria for several input signals (independently for each Input Channel) or peripheral signals, and to combine their effects to a single output, e.g. to generate an interrupt or to start an ADC conversion. This combination capability allows the generation of an interrupt per OGU that can be triggered by several inputs (multitude of request sources -> one reaction).

Pattern Detection

The pattern detection logic allows the combination of the status flags of all ETLx units. Each status flag can be individually included or excluded from the pattern detection for each OGUy, via control bits IGCRm.IPENxy. The pattern detection block outputs the following pattern detection results:

- **Pattern match** (PDRR.PDRy = 1 and ERU_PDOUTy = 1):
  A pattern match is indicated while all status flags that are included in the pattern detection are 1.
- **Pattern miss** (PDRR.PDRy = 0 and ERU_PDOUTy = 0):
  A pattern miss is indicated while at least one of the status flags that are included in the pattern detection is 0.

In addition, the pattern detection can deliver a trigger event if the pattern detection result changes from match to miss or vice-versa (if enabled by IGCRm.GEENy = 1). The

pattern result change event is logically OR-combined with the other enabled trigger events to support interrupt generation or to trigger other module functions (e.g. in an ADC). The event is indicated when the pattern detection result changes and PDRR.PDRy becomes updated.

The interrupt generation in the OGUy is based on the trigger ERU_TOUTy that can be gated (masked) with the pattern detection result ERU_PDOUTy. This allows an automatic and reproducible generation of interrupts during a certain time window, where the request event is elaborated by the trigger combination block and the time window information (gating) is given by the pattern detection. For example, interrupts can be issued on a regular time base while a combination of input signals occurs (pattern detection based on ETLx status bits).

Only four interrupt/service requests are generated by the ERU. The mapping of interrupt requests to OGUy blocks is shown in the ERU connection diagram.

A programmable gating scheme introduces flexibility to adapt to application requirements and allows the generation of interrupt requests ERU_IOUTy under different conditions:

- **Pattern match** (IGCRm.IGPy = 10_B):
  An interrupt request is issued when a trigger event occurs while the pattern detection shows a pattern match.
- **Pattern miss** (IGCRm.IGPy = 11_B):
  An interrupt request is issued when the trigger event occurs while the pattern detection shows a pattern miss.
- **Independent** of pattern detection (IGCRm.IGPy = 01_B):
  In this mode, each occurring trigger event leads to an interrupt request. The pattern detection output can be used independently from the trigger combination for gating purposes of other peripherals (independent use of ERU_TOUTy and ERU_PDOUTy with interrupt requests on trigger events).
- **No interrupts** (IGCRm.IGPy = 00_B, default setting)
  In this mode, an occurring trigger event does not lead to an interrupt request. The pattern detection output can be used independently from the trigger combination for gating purposes of other peripherals (independent use of ERU_TOUTy and ERU_PDOUTy without interrupt requests on trigger events).

8.4.1.7 ERU Output Pin Connections

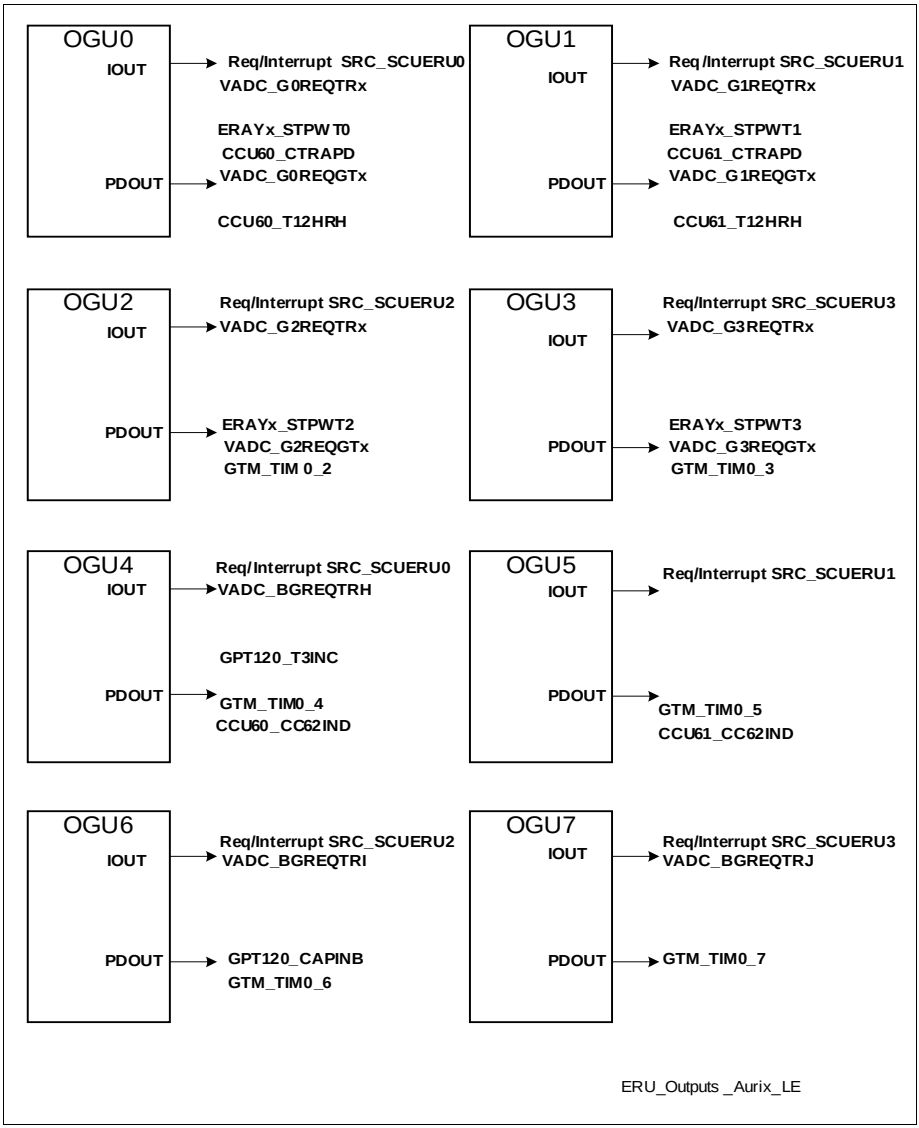


Figure 8-40 External Request Unit Output Connections for TC22x/TC23x

8.4.1.8 External Request Unit Registers

The External Input Channel Registers EICRi (i=0 to 3) for the 4 external input channels contain bits to configure the external request selection ERS and the event trigger logic ETL.

EICR0
External Input Channel Register 0 (210_H) Reset Value: $0000\ 0000_H$
EICR1
External Input Channel Register 1 (214_H) Reset Value: $0000\ 0000_H$
EICR2
External Input Channel Register 2 (218_H) Reset Value: $0000\ 0000_H$
EICR3
External Input Channel Register 3 ($21C_H$) Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	INP1			EI EN1	LD EN1	R EN1	F EN1	0	EXIS1			0			
r	rw			rw	rw	rw	rw	r	rw			r			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	INP0			EI EN0	LD EN0	R EN0	F EN0	0	EXIS0			0			
r	rw			rw	rw	rw	rw	r	rw			r			

Field	Bits	Type	Description
EXIS0	[6:4]	rw	**External Input Selection 0** This bit field determines which input line is selected for Input Channel (2i). 000_B Input (2i) 0 is selected 001_B Input (2i) 1 is selected 010_B Input (2i) 2 is selected 011_B Input (2i) 3 is selected 100_B Reserved 101_B Reserved 110_B Reserved 111_B Reserved

Field	Bits	Type	Description
FEN0	8	rw	**Falling Edge Enable 0** This bit determines if the falling edge of Input Channel (2i) is used to set bit INTF(2i). 0_B The falling edge is not used 1_B The detection of a falling edge of Input Channel 0 generates a trigger event. INTF(2i) becomes set.
REN0	9	rw	**Rising Edge Enable 0** This bit determines if the rising edge of Input Channel (2*i) is used to set bit INTF(2i). 0_B The rising edge is not used 1_B The detection of a rising edge of Input Channel (2*i) generates a trigger event. INTF(2*i) becomes set
LDEN0	10	rw	**Level Detection Enable 0** This bit determines if bit INTF(2i) is cleared automatically if an edge of the input Input Channel (2i) is detected, which has not been selected (rising edge with REN0 = 0 or falling edge with FEN0 = 0). 0_B Bit INTF(2i) will not be cleared 1_B Bit INTF(2i) will be cleared
EIEN0	11	rw	**External Input Enable 0** This bit enables the generation of a trigger event for request channel (2i) (e.g. for interrupt generation) when a selected edge is detected. 0_B The trigger event is disabled 1_B The trigger event is enabled

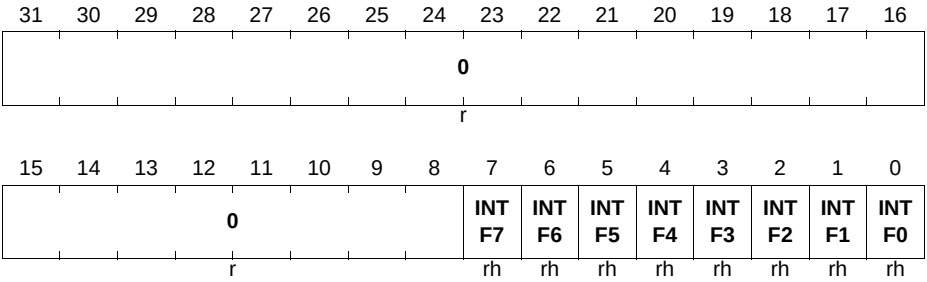
Field	Bits	Type	Description
INP0	[14:12]	rw	**Input Node Pointer** This bit field determines the destination (output channel) for trigger event (2i) (if enabled by EIEN(2i)). 000_B An event from input ETL 2i triggers output OGU0 (signal TR(2i) 0) 001_B An event from input ETL 2i triggers output OGU1 (signal TR(2i) 1) 010_B An event from input ETL 2i triggers output OGU2 (signal TR(2i) 2) 011_B An event from input ETL 2i triggers output OGU3 (signal TR(2i) 3) 100_B An event from input ETL 2i triggers output OGU4 (signal TR(2i) 0) 101_B An event from input ETL 2i triggers output OGU5 (signal TR(2i) 0) 110_B An event from input ETL 2i triggers output OGU6 (signal TR(2i) 0) 111_B An event from input ETL 2i triggers output OGU7 (signal TR(2i) 0)
EXIS1	[22:20]	rw	**External Input Selection 1** This bit field determines which input line is selected for Input Channel (2i+1). 000_B Input (2i+1) 0 is selected 001_B Input (2i+1) 1 is selected 010_B Input (2i+1) 2 is selected 011_B Input (2i+1) 3 is selected 100_B Reserved 101_B Reserved 110_B Reserved 111_B Reserved
FEN1	24	rw	**Falling Edge Enable 1** This bit determines if the falling edge of Input Channel (2i+1) is used to set bit INTF(2i+1). 0_B The falling edge is not used 1_B The detection of a falling edge of Input Channel 1 generates a trigger event. INTF(2i+1) becomes set

Field	Bits	Type	Description
REN1	25	rw	**Rising Edge Enable 1** This bit determines if the rising edge of Input Channel (2i+1) is used to set bit INTF(2i+1). 0_B The rising edge is not used 1_B The detection of a rising edge of Input Channel 1 generates a trigger event . INTF(2i+1) becomes set
LDEN1	26	rw	**Level Detection Enable 1** This bit determines if bit INTF(2i+1) is cleared automatically if an edge of the input Input Channel (2i+1) is detected, which has not been selected (rising edge with REN1 = 0 or falling edge with FEN1 = 0). 0_B Bit INTF(2i+1) will not be cleared 1_B Bit INTF1(2i+1) will be cleared
EIEN1	27	rw	**External Input Enable 1** This bit enables the generation of a trigger event for request channel (2i+1) (e.g. for interrupt generation) when a selected edge is detected. 0_B The trigger event is disabled 1_B The trigger event is enabled
INP1	[30:28]	rw	**Input Node Pointer** This bit field determines the destination (output channel) for trigger event (2i+1) (if enabled by EIEN(2i+1)). 000_B An event from input ETL 2i+1 triggers output OGU0 (signal TR(2i+1) 0) 001_B An event from input ETL 2i+1 triggers output OGU1 (signal TR(2i+1) 1) 010_B An event from input ETL 2i+1 triggers output OGU2 (signal TR(2i+1) 2) 011_B An event from input ETL 2i+1 triggers output OGU3 (signal TR(2i+1) 3) 100_B An event from input ETL 2i+1 triggers output OGU4 (signal TR(2i+1) 0) 101_B An event from input ETL 2i+1 triggers output OGU5 (signal TR(2i+1) 0) 110_B An event from input ETL 2i+1 triggers output OGU6 (signal TR(2i+1) 0) 111_B An event from input ETL 2i+1 triggers output OGU7 (signal TR(2i+1) 0)

Field	Bits	Type	Description
0	[3:0], 7, [19:15], 23, 31	r	**Reserved** Read as 0; should be written with 0.

The External Input Flag Register EIFR contains all status flags for the external input channels. The bits in this register can be cleared by software by setting FMR.FCx, and set by setting FMR.FSx.

EIFR
External Input Flag Register **(220_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
INTFx (x = 0-7)	x	rh	**External Event Flag of Channel x** This bit monitors the status flag of the event trigger condition for the input channel x. This bit is automatically cleared when the selected condition (see RENx, FENx) is no longer met (if LDENx = 1) or remains set until it is cleared by software (if LDENx = 0).
0	[31:8]	r	**Reserved** Read as 0; should be written with 0.

FMR
Flag Modification Register **(224_H)** **Reset Value: 0000 0000$_H$**

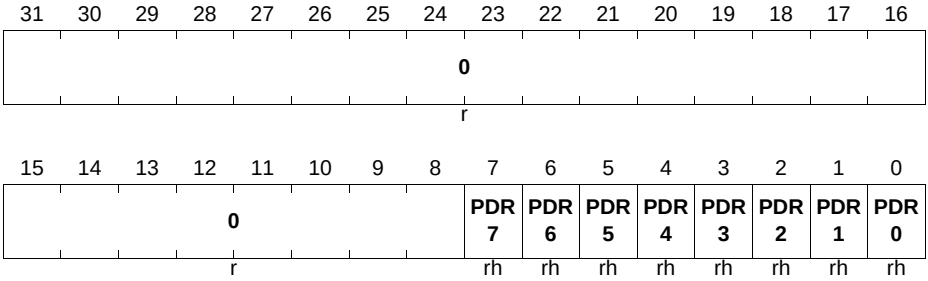
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								FC7	FC6	FC5	FC4	FC3	FC2	FC1	FC0
r								w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								FS7	FS6	FS5	FS4	FS3	FS2	FS1	FS0
r								w	w	w	w	w	w	w	w

Field	Bits	Type	Description
FSx (x = 0-7)	x	w	**Set Flag INTFx for Channel x** Setting this bit will set the corresponding bit INTFx in register EIFR. Reading this bit always delivers a 0. If both FSx and FCx are set in the same access then the bit x in register EIFR is not modified. 0_B The bit x in register EIFR is not modified 1_B The bit x in register EIFR is set
FCx (x = 0-7)	16 + x	w	**Clear Flag INTFx for Channel x** Setting this bit will clear the corresponding bit INTFx in register EIFR. Reading this bit always delivers a 0. If both FSx and FCx are set in the same access then the bit x in register EIFR is not modified. 0_B The bit x in register EIFR is not modified 1_B The bit x in register EIFR is cleared
0	[15:8], [31:24]	r	**Reserved** Read as 0; should be written with 0.

The Pattern Detection Result Register monitors the combinatorial output status of the pattern detection units.

PDRR
Pattern Detection Result Register (228_H) Reset Value: 0000 00FF$_H$



Field	Bits	Type	Description
PDRy (y = 0-7)	y	rh	**Pattern Detection Result of Channel y** This bit monitors the output status of the pattern detection for the output channel y.
0	[31:8]	r	**Reserved** Read as 0; should be written with 0.

The Interrupt Gating Control Registers IGCRj (j=0 to 3) contain bits to enable the pattern detection and to control the gating for the output channels.

IGCR0
Flag Gating Register 0 **($22C_H$)** **Reset Value: 0000 0000$_H$**
IGCR1
Flag Gating Register 1 **(230_H)** **Reset Value: 0000 0000$_H$**
IGCR2
Flag Gating Register 2 **(234_H)** **Reset Value: 0000 0000$_H$**
IGCR3
Flag Gating Register 3 **(238_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IGP1		GEEN1	0					IPEN17	IPEN16	IPEN15	IPEN14	IPEN13	IPEN12	IPEN11	IPEN10
rw		rw	r					rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IGP0		GEEN0	0					IPEN07	IPEN06	IPEN05	IPEN04	IPEN03	IPEN02	IPEN01	IPEN00
rw		rw	r					rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
IPEN0x (x = 0-7)	x	rw	**Flag Pattern Enable for Channel 0** Bit IPEN0x determines if the flag INTFx of channel x takes part in the pattern detection for the gating of the requests for the output signals GOUT(2*j) and IOUT(2*j). 0_B The bit INTFx does not take part in the pattern detection 1_B The bit INTFx is taken into consideration for the pattern detection
GEEN0	13	rw	**Generate Event Enable 0** Bit GEEN0 enables the generation of a trigger event for output channel (2*j) when the result of the pattern detection changes. When using this feature, a trigger (e.g. for an interrupt) is generated during the first clock cycle when a pattern is detected or when it is no longer detected. 0_B The trigger generation at a change of the pattern detection result is disabled 1_B The trigger generation at a change of the pattern detection result is enabled

Field	Bits	Type	Description
IGP0	[15:14]	rw	**Interrupt Gating Pattern 0** In each register IGCRj, bit field IGP0 determines how the pattern detection influences the output lines GOUT(2j) and IOUT(2j). 00_B IOUT(2j) is inactive.The pattern is not considered. 01_B IOUT(2j) is activated in response to a trigger event. The pattern is not considered. 10_B The detected pattern is considered. IOUT(2j) is activated if a trigger event occurs while the pattern is present. 11_B The detected pattern is considered. IOUT(2j) is activated if a trigger event occurs while the pattern is not present.
IPEN1x (x = 0-7)	16+x	rw	**Interrupt Pattern Enable for Channel 1** Bit IPEN(2j+1)x determines if the flag INTFx of channel (2j+1) takes part in the pattern detection for the gating of the requests for the output signals GOUT(2j+1) and IOUT(2j+1). 0_B The bit INTFx does not take part in the pattern detection 1_B The bit INTFx is taken into consideration for the pattern detection
GEEN1	29	rw	**Generate Event Enable 1** Bit GEEN1 enables the generation of a trigger event for output channel (2j+1) when the result of the pattern detection changes. When using this feature, a trigger (e.g. for an interrupt) is generated during the first clock cycle when a pattern is detected, or when it is no longer detected. 0_B The trigger generation at a change of the pattern detection result is disabled 1_B The trigger generation at a change of the pattern detection result is enabled

Field	Bits	Type	Description
IGP1	[31:30]	rw	**Interrupt Gating Pattern 1** In each register IGCRj, bit field IGP1 determines how the pattern detection influences the output lines GOUT(2j+1) and IOUT(2j+1). 00_B IOUT(2j+1) is inactive.The pattern is not considered. 01_B IOUT(2j+1) is activated in response to a trigger event. The pattern is not considered. 10_B The detected pattern is considered. IOUT(2j+1) is activated if a trigger event occurs while the pattern is present. 11_B The detected pattern is considered. IOUT(2j+1) is activated if a trigger event occurs while the pattern is not present.
0	[12:8], [28:24]	r	**Reserved** Read as 0; should be written with 0.

8.4.2 Lockstep CPU Configuration

This section contains the registers which are used to configure and enable CPU Lockstep Mode. These registers are only writeable by start-up firmware and are configured from the BMI . See the Firmware and Lockstep Control Logic chapters for further details.

8.4.2.1 Logic Monitor Control Registers

LCL CPU0 Control

Provides control for CPU0 lockstep compare logic block.

LCLCON0
LCL CPU0 Control Register **(134_H)** **Reset Value: 8001 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LSEN							0								LS
rw							r								rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
0	[15:0]	r	**Reserved** will be read as 0_B, should be written as 0_B
LS	16	rh	**Lockstep Mode Status** This bit indicates whether CPU0 is currently running in lockstep monitor mode 0_B Not in lockstep mode 1_B Running in lockstep mode
0	[30:17]	r	**Reserved** will be read as 0_B, should be written as 0_B
LSEN	31	rw	**Lockstep Enable** This bit may only be written by SSW during boot. Enable lockstep CPU monitoring for the associated processor core, CPU0. After cold reset, lockstep is enabled by default. The LSEN bit may be cleared during the boot to disable lockstep mode. SMU lockstep fault reporting should be disabled when lockstep is disabled. 0_B Lockstep is disabled 1_B Lockstep enabled (Default after Cold Power-On Reset)

LCL CPU1 Control

Provides control for CPU1 lockstep compare logic block.

LCLCON1
LCL CPU1 Control Register (138$_H$) Reset Value: 8000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LSEN							0								LS
rw							r								rh

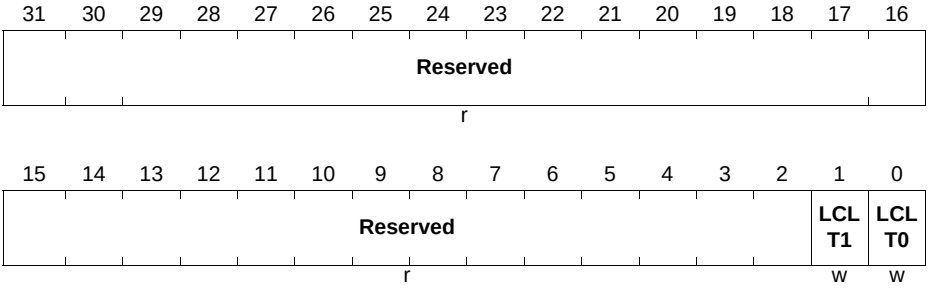
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
0	[15:0]	r	**Reserved** will be read as 0_B, should be written as 0_B
LS	16	r	**Reserved in this product** will be read as 0_B, should be written as 0_B
0	[30:17]	r	**Reserved** will be read as 0_B, should be written as 0_B
LSEN	31	rw	**Reserved in this product** will be read as 1_B, should be written as 1_B

LCL Test Register

Provides the capability for software to inject a fault condition into the comparators of each lockstep compare logic block. The implementation should generate a single cycle fault each time the bit is written with ‘1’.

LCLTEST
LCL Test Register **($13C_H$)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
0	[31:2]	r	**Reserved**
LCLT0	0	rwh	**LCL0 Lockstep Test** Fault injection for LCL0. Reads as zero. 0_B No action 1_B Inject single fault in LCL0
LCLT1	1	r	**Reserved in this product**

Critical Register Monitoring

Safety critical SCU registers are protected by the Safety Endinit signal against accidental software writes. In addition to this, a continuous monitoring mechanism is provided to check the content of these safety critical registers have not been affected by random single bit flip-flop failures. This mechanism is controlled by the SMU.

The System Control registers with write access mode “SE” listed in **Table 8-29** are covered by this mechanism. This includes:

- CCU registers OSCCON, PLLCONx, PLLERAYCONx, CCUCONx, FDR, EXTCON
- RCU registers RSTCON, ARSTDIS, SWRSTCON, RSTCONx, ESRCFGx, ESROCFG, PDISC
- EVR registers EVRRSTCON, EVRSDCOEFF*, EVRSDCTRL*, EVRTRIM, EVR13CON
- PMC registers PMSWCRx, PMCSRx , PMSWSTATCLR
- Emergency Stop register EMSR
- Registers controlling safety monitors WDTSCONx1, WDTCPU*CON1, LBISTCTRLx, EVRUVMON, EVROVMON
- MTU registers MEMTEST, MEMMAP, MEMSTAT, ECCS and RAR

Any single bit flipflop failures will be reported via an SMU alarm.

8.4.3 Die Temperature Measurement

The Die Temperature Sensor (DTS) generates a measurement result that indicates directly the current temperature. The result of the measurement is displayed via bit field DTSSTAT.RESULT. In order to start one measurement bit DTSCON.START needs to be set.

The DTS has to be enabled before it can be used via bit DTSCON.PWD. When the DTS is powered after the start-up time of the DTS (defined in the Data Sheet) a temperature measurement can be started.

Note: If bit field DTSSTAT.RESULT is read before the first measurement was finished it will return 0x000.

When a measurement is started the result is available after the measurement time passed. If the DTS is ready to start a measurement can be checked via bit DTSSTAT.RDY. If a started measurement is finished or still in progress is indicated via the status bit DTSSTAT.BUSY. The measurement time is also defined in the Data Sheet.

In order to adjust production variations bit field DTSCON.CAL is available. DTSCON.CAL is automatically programmed with a predefined value.

Note: The first measurement after the DTS was powered delivers a result without calibration adjustment and should be ignored therefore.

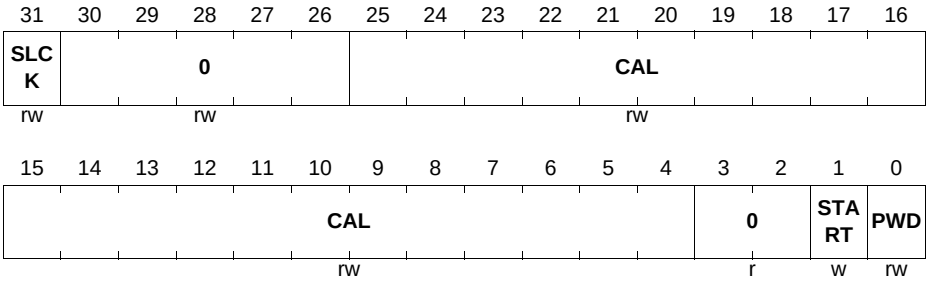
The formula to calculate the die temperature is defined in the Data Sheet.

Note: The maximum resolution is only achieved for a measurement that is part of multiple continuous measurements (recommended to ignore are two here).

An interrupt service request (SRC_SCUDTS) is generated when DTS busy indication changes from 1 to 0.

8.4.3.1 Die Temperature Sensor Register

DTSCON
Die Temperature Sensor Control Register($0E4_H$) **Reset Value: $0000\ 0001_H$**



Field	Bits	Type	Description
PWD	0	rw	**Sensor Power Down** This bit defines the DTS power state. 0_B The DTS is powered 1_B The DTS is not powered
START	1	w	**Sensor Measurement Start** This bit starts a measurement of the DTS. 0_B No DTS measurement is started 1_B A DTS measurement is started If set this bit is automatically cleared. This bit always reads as zero.
CAL	[25:4]	rw	**Calibration Value** This bit field interfaces the calibration values to the DTS.

Field	Bits	Type	Description
SLCK	31	rw	**Security Lock** 0_B No lock active 1_B Lock is active If this bit is set, no bits in this register (except START) are writeable. Attempted writes when LCK is set will cause an SMU SPB error alarm trigger. This bit can not by cleared by software. This bit can only be set by an access from the HSM master (TAG = 001101_B). A set operation performed by any other master is ingnored and the bit is not modified.
0	[3:2], [30:24]	r	**Reserved** Read as 0; should be written with 0.

DTSSTAT

Die Temperature Sensor Status Register($0E0_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	RDY	0				RESULT									
rh	rh	r				rh									

Field	Bits	Type	Description
RESULT	[9:0]	rh	**Result of the DTS Measurement** This bit field shows the result of the DTS measurement. The value given is directly related to the die temperature. Only the bit [9:2] have to be evaluated.
RDY	14	rh	**Sensor Ready Status** This bit indicate the DTS is ready or not. 0_B The DTS is not ready 1_B The DTS is ready

Field	Bits	Type	Description
BUSY	15	rh	**Sensor Busy Status** This bit indicate the DTS is currently busy or not. If the sensor is busy currently a measurement is running and the result should not be used. 0_B The DTS is not busy 1_B The DTS is busy *Note: This bit is updated 2 cycles after bit DTSCON.START is set.*
0	[13:10], [31:16]	r	**Reserved** Read as 0; should be written with 0.

DTSLIM

Die Temperature Sensor Limit Register(240_H) **Reset Value: $03FF\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UOF	**SLCK**	**0**				**UPPER**									
rwh	rw	r				rw									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LLU	**0**					**LOWER**									
rwh	r					rw									

Field	Bits	Type	Description
LOWER	[9:0]	rw	**Lower Limit** This bit field defines the lower limit of the DTS temerature check. The DTS measurement result is compared against this value and if the measurement result is smaller than LOWER bit LLU is set.

Field	Bits	Type	Description
LLU	15	rwh	**Lower Limit Underflow** 0_B No temperature underflow was detected 1_B A temperature underflow was detected When this bit is set an HSM temperature underflow trigger is generated. When this bit is set an temperature SMU DTS alarm trigger is generated. This bit is cleared by writting a zero. Writting a one has no effect. This bit is updated when either a DTS measurement is finished or if bit field LOWER is changed.
UPPER	[25:16]	rw	**Upper Limit** This bit field defines the upper limit of the DTS temperature check. The DTS measurement result is compared against this value and if the measurement result is greater than UPPER bit UOF is set
SLCK	30	rw	**Security Lock** 0_B No lock active 1_B Lock is active If this bit is set both bit fields LOWER and UPPER can not longer be written. Write requests when LCK is set will cause a SMU SPB error alarm trigger. This bit can not by cleared by software. This bit can only be set by an access from the HSM master (TAG = 001101_B). A set operation performed by any other master is ingnored and the bit is keep as cleared.
UOF	31	rh	**Upper Limit Overflow** 0_B No temperature overflow was detected 1_B A temperature overflow was detected When this bit is set an HSM temperature overflow trigger is generated. When this bit is set an SMU DTS alarm trigger is generated. This bit is cleared by writting a zero. Writting a one has no effect. This bit is updated when either a DTS measurement is finished or if bit field UPPER is changed.

Field	Bits	Type	Description
0	[14:10], [29:26]	r	**Reserved** Read as 0; should be written with 0.

8.4.4 Watchdog Timers

This section describes the TC21x/TC22x/TC23x Watchdog Timers (WDTs). Topics include an overview of the WDT functions and descriptions of the registers, the password-protection scheme, accessing registers, modes, and initialization.

The TC21x/TC22x/TC23x contains the following Watchdog Timers:

- Safety Watchdog
- CPU0 Watchdog

8.4.4.1 Watchdog Timers Overview

The WDTs provide a highly reliable and secure way to detect and recover from software or hardware failure. The WDTs help to abort an accidental malfunction of a CPU or TC21x/TC22x/TC23x system within a user-specified time period.

In addition to this standard “Watchdog” function, each of the WDTs incorporates an End-of-Initialization (ENDINIT) feature which can protect critical registers from accidental writes.

Servicing the Watchdogs and modifying the ENDINIT bits are critical functions that must not be allowed in case of a system malfunction. To protect these functions a sophisticated scheme is implemented that requires a password and guard bits during accesses to the WDT control registers. Any write access that does not deliver the correct password or the correct value for the guard bits is regarded as a malfunction of the system, and results in a watchdog alarm. In addition, even after a valid access has been performed and an ENDINIT bit has been cleared to provide access to the critical registers, the Watchdog imposes a time limit for this access window. If the ENDINIT bit has not been properly set again before this limit expires, the system is assumed to have malfunctioned, and a Watchdog alarm is reported to the Safety Management Unit (SMU). These stringent requirements, although not guaranteed, nonetheless provide a high degree of assurance of the robustness of system operation.

Configuration options are available which enable a Watchdog service to additionally check code execution sequence and intermediate code execution time. If these checks are enabled then any incorrect sequence or out-of-limit execution time will also result in an SMU alarm request.

Expiry of any of the TC21x/TC22x/TC23x WDTs leads to an SMU alarm request. It is possible to program the SMU to provide an interrupt or NMI to provide some time for recovery or status recording before further action is taken (eg reset of the device or stopping of the CPU via idle request).

Safety Watchdog

The Safety Watchdog Timer provides an overall system level watchdog which is independent from the CPU watchdogs and also provides temporal protection against unintended writes to critical system registers which could have impact on safety-critical

systems. When the Safety WDT is enabled, it can cause an SMU alarm request if it is not serviced within a user-programmable time period. A CPU must service the Safety WDT within this time interval to prevent this. The response to a Safety WDT timeout is configurable within the SMU. Hence, periodic servicing of the Safety WDT confirms that the system is functioning as expected.

Typically the SCU write protection (ACCEN) will be configured so that only restricted “Safety” CPU(s) can configure system-critical functionality. This includes the ability to service the Safety Watchdog. In addition, Safety Watchdog disable/enable/configuration function requires a Safety ENDINIT password.

The registers marked “SE” in **Table 8-29** and other module Register Tables are considered to be static properties of a safety-critical system and are all write-protected by the Safety ENDINIT feature so that they cannot be changed without unlocking the Safety Watchdog.

CPU Watchdogs

The individual CPU Watchdog Timers provide the ability to monitor separate CPU execution threads without the need for software to co-ordinate the shared use of a common watchdog. Each CPUx watchdog timer also incorporates a local CPUx_Endinit feature which provides temporal protection against unintended writes to critical local CPU registers and also some system registers which require protection but are not already covered by the Safety ENDINIT.

When a CPU WDT is enabled it can cause an SMU alarm request if it is not correctly serviced within a user-programmable time period. The CPU must service its CPU WDT within this time interval to prevent this. The response to each CPUx watchdog timeout is configurable within the SMU. Hence, periodic servicing of a CPU WDT confirms that the corresponding CPU is executing a software sequence as expected.

After a reset, CPU0 runs and CPU0 watchdog timer starts automatically. Other CPUs are initially in a HALT state and therefore their corresponding Watchdog Timers are disabled. The other CPU Watchdogs are not configured to generate a time-out reset by default, but this can be enabled (See SMU section). A CPU Watchdog may only be configured, enabled or disabled by its corresponding CPU.

8.4.4.2 Features of the Watchdog Timers

The main features of the WDTs are summarized here.

- 16-bit Watchdog counter
- Selectable input frequency: f_{SPB}/64, f_{SPB}/256 or f_{SPB}/16384
- 16-bit user-definable reload value for normal Watchdog operation, fixed reload value for Time-Out Mode
- Incorporation of the corresponding ENDINIT bit and monitoring of its modifications
- Sophisticated Password Access mechanism with user-definable password fields
- Access Error Detection: Invalid password (during first access) or invalid guard bits (during second access) trigger an alarm request to the SMU
- Temporal and logical monitoring capabilities:
  - Optional Code Sequence checking. An incorrect code sequence signature identification will trigger an alarm request to the SMU
  - Optional Code Execution Time checking. Code execution times out of expected limits will trigger an alarm request to the SMU.
- Overflow Error Detection: An overflow of the WDT counter triggers an alarm request to the SMU
- Watchdog function can be disabled; access protection and ENDINIT bit monitor function remain enabled
- Configurable mechanism to prevent Watchdog reloads after an unserviced safety warning alarm to ensure that unserviced warnings lead to an SMU hardware response
- External “Alive heartbeat” indication to show that WDT is running and SMU is not in alarm mode

8.4.4.3 The Endinit Functions

There are a number of registers in the TC21x/TC22x/TC23x that are usually programmed only once during the initialization sequence of the system or application. Modification of such registers during normal application run can have a severe impact on the overall operation of modules or the entire system.

While the Supervisor Mode and the Access Protection scheme provide a certain level of protection against unintentional modifications, they may not be sufficient to protect against all unintended accesses to system-critical registers.

The TC21x/TC22x/TC23x provides one more level of protection for such registers via the various Endinit features. This is a highly secure write-protection scheme that makes unintentional modifications of registers protected by this feature nearly impossible.

The Endinit feature consists of an ENDINIT bit incorporated in each WDT control register. Registers protected via an Endinit determine whether or not writes are enabled. Writes are only enabled if the corresponding ENDINIT = 0 AND Supervisor Mode is active. Write attempts if this condition is not true will be discarded and the register

contents will not be modified in this case. The BCU controls the further operation following a discarded write access.

To get the highest level of security, writes to the ENDINIT bits are protected by a highly secure access protection scheme implemented in the WDTs. This is a complex procedure, that makes it nearly impossible for ENDINIT bits to be modified unintentionally. In addition, each WDT monitors ENDINIT bit modifications by starting a time-out sequence each time software opens access to the critical registers through clearing the corresponding ENDINIT bit. If the time-out period ends before the corresponding ENDINIT bit is set again, a malfunction of the software is assumed and a Watchdog fault response is generated.

The access-protection scheme and the Endinit time-out operation of the WDTs is described in the following sections. In the register overview tables for each module (including the SCU itself) the registers which are protected via each Endinit type are identified in the column describing write accesses as follows:

- “CE0”- writeable only when CPU0 ENDINIT bit is zero
- “E” - writeable when any (one or more) CPUx ENDINIT bit is zero
- “SE” - writeable only when Safety ENDINIT bit is zero
- None of the above - accessible at any time

Note: The clearing of an ENDINIT bit takes some time. Accesses to Endinit-protected registers after the clearing of an ENDINT bit must only be done when the ENDINIT bit is really cleared. As a solution the ENDINIT bit should be read back once before Endinit-protected registers are accessed the first time after bit ENDINIT has been cleared.

Password Access to WDTxCON0

A correct password must be written to register WDTxCON0 (x=S,CPU0) in order to unlock it for modifications. Software must either know the correct password in advance or compute it at runtime. The passwords for each of the Watchdogs (x=S,CPU0) can be different in order to provide independent watchdog functionality program flows to have independent watchdog functions.

The Safety Watchdog password register WDTSCON0 is protected by the generic SCU protection scheme which allows only configured master(s) to have write access (See **ACCEN0**).

CPU-specific Watchdog password registers WDTCPUyCON0 are individually protected such that they may only be written by the corresponding CPUy

A watchdog may be used within a safety application to provide a recovery time period during which software might attempt to recover from a safety alarm warning. To ensure that a CPU fault could not allow a fault to be ignored an option is provided to prevent watchdog unlocking if the Safety Management Unit (SMU) is not in the RUN state. This option may be enabled by bit WDTxCON0.UR.

If the password is valid and the SMU state meets the requirements of the WDTxCON0.US bit then WDTxCON0 will be unlocked as soon as the Password Access is completed. The unlocked condition will be indicated by WDTxCON0.LCK = 0. To ensure the correct servicing sequence, a password access is only permitted when the WDTxCON0.LCK bit was set prior to the access.

If an improper password value is written to WDTxCON0 during the Password Access, a Watchdog Access Error condition exists. Bit WDTxSR.AE is set and an alarm request is sent to the Safety Management Unit (SMU).

The 14-bit user-definable password, WDTxCON0.PW, provides additional options for adjusting the password requirements to the application’s needs. It can be used, for instance, to detect unexpected software loops, or to monitor the execution sequence of routines.

Table 8-25 summarizes the requirements for the password. Various options exist, which are described in more detail below

Table 8-25 Password Access Bit Pattern Requirements

Bit Position	Required Value
[1:0]	Fixed; must be written to 01_B
[15:2]	If PAS=0: WDTxCON0.PW[7:2] must be written with inverted current value read from WDTxCON0.PW[7:2] WDTxCON0.PW[15:8] must be written with non-inverted current value read from WDTxCON0.PW[15:8] If PAS=1: Must be written with Expected Next Sequence Password
[31:16]	If TCS=0: Must be written with current value of user-definable reload value, WDTxCON0.REL If TCS=1: Must be written with inverted estimate of the WDT count value, WDTxSR.TIM. This value must be within +/- WDTxSR.TCT of the actual value

Static Password

In the static password mode (WDTxSR.PAS=0) the password can only be changed by a valid Modify Access. The Password Access has been designed such that it is not possible simply to read the register and re-write it. Some of the password read bits must be inverted (toggled) before being re-written. This prevents a simple malfunction from accidentally unlocking the WDT through a simple read/write sequence.

Automatic Password Sequencing

If automatic password sequencing is enabled (WDTxSR.PAS=1) then the password changes automatically after each password check (i.e. Password Access or Check Access). The Expected Next Password follows a pseudo-random sequence based upon a 14-bit Fibonacci LFSR (Linear Feedback Shift Register) with characteristic polynomial x14+x13+x12+x2+1. An initial password (or subsequent manual password updates) can also be provided by Modify Accesses.

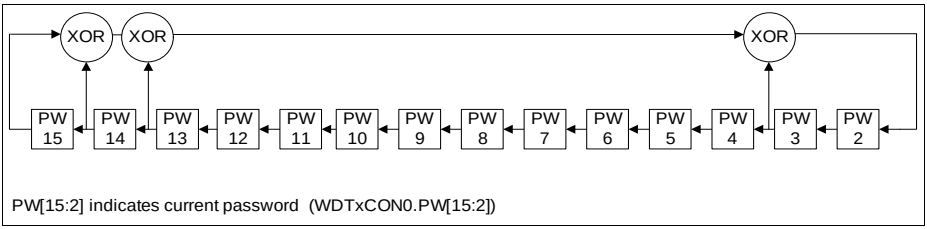


Figure 8-41 Password Sequencing LFSR

Time-Independent Password

If time checking is not enabled (WDTxSR.TCS=0) then the REL field of the WDTxCON0 register must be simply re-written with the existing reload value during the Password Access.

Time Check Password

If time checking is enabled (WDTxSR.TCS=1) the the REL field of the WDTxCON0 register must be written with an inverted (bitflipped) estimate of the current WDT count value. The acceptable margin of error of this estimate (in WDT clock periods) is specified by the value of WDTxSR.TCT. If the written estimate is outside the range WDTxSR.TIM +/- WDTxSR.TCT, then an SMU alarm condition is indicated. This mechanism can provide a check of the elapsed program execution time since the last WDT restart. Note that a Time Check comparison is still required for a Password or Check Access while the WDT is operating in Time-Out mode (e.g. After accessing ENDINIT-protected registers).

Check Access to WDTxCON0

A Check Access is identical to a Password Access except that the lock bit is not cleared and a subsequent Modify Access is therefore not allowed. A Check Access does not trigger an SMU Alarm Request provided that the write data requirements are satisfied. A Check Access may only be performed when the LCK bit is set.

This type of access is used for intermediate checkpoints between WDT services. This may be used (e.g. in conjunction with the timestamp count checking feature or sequence passwords) for task sequence or execution time monitoring.

Table 8-26 Check Access Bit Pattern Requirements

Bit Position	Required Value
[1:0]	Fixed; must be written to 11_B
[15:2]	If PAS=0: WDTxCON0.PW[7:2] must be written with inverted current value read fromWDTxCON0.PW[7:2] WDTxCON0.PW[15:8] must be written with non-inverted current value read from WDTxCON0.PW[15:8] If PAS=1: Must be written with Expected Next Sequence Password
[31:16]	If TCS=0: Must be written with current value of user-definable reload value, WDTxCON0.REL If TCS=1: Must be written with inverted estimate of the WDT counter WDTxSR.TIM. This value must be within +/- WDTxSR.TCT of the actual value

If an improper value is written to WDTxCON0 during the Check Access, a Watchdog Access Error condition exists. Bit WDTxSR.AE is set and an alarm request is sent to the Safety Management Unit (SMU).

Modify Access to WDTxCON0

If a WDTxCON0 is successfully unlocked by a Password Access, the following write access to WDTxCON0 can modify it. However, this access must also meet certain requirements in order to be accepted and regarded as valid. **Table 8-27** lists the required bit patterns. If the access does not follow these rules, a Watchdog Access Error condition is detected, bit WDTxSR.AE is set, and an alarm request is sent to the Safety Management Unit (SMU).

Table 8-27 Modify Access Bit Pattern Requirements

Bit Position	Value
0	User-definable; desired value for bit WDTxCON0.ENDINIT.
1	Fixed; must be written with 1_B.
[15:2]	User-definable; desired value of user-definable password field, WDTxCON0.PW.
[31:16]	User-definable; desired value of user-definable reload value, WDTxCON0.REL.

After the Modify Access has completed, WDTxCON0.LCK is set again, automatically re-locking WDTxCON0. Before the register can be modified again, a valid Password Access must be executed again.

External WDT “Alive Heartbeat” Indication

The current state of the SAFECON.HBT bit may optionally be routed to external pin(s). This register may only be written during a Safety WDT service (between a Modify Access with Safety ENDINIT=0 and the subsequent Modify Access with Safety ENDINIT=1). Periodic toggling of this bit by software can therefore be used to indicate to an external monitor that the CPU is operational and the Safety Watchdog Timer is still running. If the Unlock Restriction (UR bit) is also enabled then any other SMU safety alarm condition will prevent a WDT refresh and hence stop the heartbeat automatically.

Access to Endinit-Protected Registers

If write access to Endinit-protected registers is required during run time, write access can be re-enabled for a limited time period. To re-enable access, WDTxCON0 must first be unlocked with a valid Password Access. In the subsequent valid Modify Access, ENDINIT can be cleared. Access to Endinit-protected registers is now open again. However, when WDTxCON0 is unlocked, the WDT is automatically switched to Time-Out Mode. The access window is therefore time-limited. Time-Out Mode is only terminated after ENDINIT has been set again, requiring another Valid Password and Valid Modify Access to WDTxCON0.

If the WDT is not used in an application and is therefore disabled (WDTxSR.DS = 1), the above described case is the only occasion when WDTxCON0 must be accessed again after the system is initialized. If there are no further changes to critical system registers needed, no further accesses to WDTxCON0, WDTxCON1, or WDTxSR are necessary. However, it is recommended that the WDT be used in an application for safety reasons.

For debugging support the Cerberus module can override all the ENDINIT controls of all WDTs to ease the debug flow. If bit CBS_OSTATE.ENIDIS is set all ENDINIT protection is disabled independent of the current status configured by the WDTs. If CBS_OSTATE.ENIDIS is cleared the complete control is within the WDTs.

8.4.4.4 Timer Operation

The timers for Safety Watchdog and CPU0 are automatically active after an Application Reset . Each 16-bit counter implementing the timer functionality is either triggered with f_{SPB} / 64, f_{SPB} / 256 or f_{SPB} / 16384. The three possible counting rates are controlled via bit WDTxCON1.IRx.

Determining WDT Periods

All WDTs use the SPB clock f_{SPB}. A clock divider in front of each WDT provides three WDT counter frequencies, f_{SPB} / 64, f_{SPB} / 256 and f_{SPB} / 16384.

The general formula to calculate a Watchdog timeout period is:

(8.26)

$$\text{period} = \frac{(2^{16} - \text{startvalue}) \cdot \text{divider}}{f_{SPB}}$$

The parameter startvalue represents the fixed value $FFFC_H$ for the calculation of the Time-Out Period, and the user-programmable reload value WDTxCON0.REL for the calculation of the Normal Period.

The parameter divider represents the user-programmable source clock division selected by WDTxCON0.IRx, which can be 64, 256 or 16384.

Timer Modes

Each Watchdog Timer can operate in one of three different operating modes:

- Time-Out Mode
- Normal Mode
- Disable Mode

The following overview describes these modes and how the WDT changes from one mode to the other.

Time-Out Mode

The Time-Out Mode is entered after an Application Reset or when a valid Password Access to register WDTxCON0 is performed. The Time-Out Mode is indicated by bit WDTxSR.TO = 1. The timer is set to $FFFC_H$ and starts counting upwards. Time-Out Mode can only be exited properly by setting ENDINIT = 1 with a correct access sequence. If an improper access to the WDT is performed, or if the timer overflows before ENDINIT is set, and an alarm request is sent to the Safety Management Unit (SMU).

A proper exit from Time-Out Mode can either be to the Normal or the Disable Mode, depending on the state of the disable request bit WDTxCON1.DR.

Normal Mode

In Normal Mode (DR = 0), the WDT operates in a standard Watchdog fashion. The timer is set to WDTxCON0.REL, and begins counting up. It has to be serviced before the counter overflows. Servicing is performed through a proper access sequence to the control register WDTxCON0. This enters the Time-Out Mode.

If the WDT is not serviced before the timer overflows, a system malfunction is assumed. Normal Mode is terminated and an alarm request is sent to the Safety Management Unit (SMU)

Disabled Mode

Disabled Mode is provided for applications which truly do not require the WDT function. It can be requested from Time-Out Mode when the disable request bit WDTxCON1.DR is set. The Disabled Mode is entered when was requested AND bit WDTxCON0.ENDINIT is set. The timer is stopped in this mode. However, disabling the WDT only stops it from performing the standard Watchdog function, eliminating the need for timely service of the WDT. It does not disable Time-Out mode. If an access to register WDTxCON0 is performed in Disabled Mode, Time-Out Mode is entered if the access was valid, and an alarm request is sent to the Safety Management Unit (SMU) if the access was invalid. Thus, the ENDINIT monitor function as well as (a part of) the system malfunction detection will still be active.

WDT Alarm Request

SMU alarm requests are generated for three cases:

- Invalid write data during access to register WDTxCON0
- Not executing a password access before a timer overflow occurs in the Time-Out Mode
- Not serving the WDT before a timer overflow occurs in the Normal Mode

The resulting alarm response(eg NMI, Reset etc) is programmable within the Safety Management Unit (SMU).

Note: The WDT itself is reset by any Application Reset .

Servicing the Watchdog Timer

If the WDT is used in an application and is enabled (WDTxSR.DS = 0), it must be regularly serviced to prevent it from overflowing.

Service is performed in two steps, a valid Password Access followed by a valid Modify Access. The valid Password Access to WDTxCON0 automatically switches the WDT to Time-Out Mode. Thus, the Modify Access must be performed before the time-out expires or an alarm request will be sent to the Safety Management Unit (SMU).

During the next Modify Access, the strict requirement is that WDTxCON0.ENDINIT is written with 1.

Note: ENDINIT must be written with 1 to perform a proper service, even if it is already set to 1.

Changes to the reload value WDTxCON0.REL, or the user-definable password WDTxCON0.PW, are not required.

When a WDT service is properly executed, Time-Out Mode is terminated, and the WDT switches back to its former mode of operation, and the WDT service is complete.

Check Accesses are optional and may be performed at any time when the WDT is locked.

WDT Operation During Power-Saving Modes

If one or more of the CPU are in Idle Mode, they cannot service a WDT because no software is running. Excluding the case where the system is running normally, a strategy for managing WDTs is needed while a CPU is in Idle Mode. There are two ways to manage a WDT in these cases. Firstly, the Watchdog can be disabled before idling the CPU. The disadvantage of this is that the system will no longer be monitored during the idle period.

A better approach to this problem relies upon the wake-up feature of the WDTs. Whenever CPUx is put in Idle or Sleep Mode and the WDT is not disabled, the CPUx is woken at regular intervals. When WDTx changes its count value (WDTxSR.TIM) from $7FFF_H$ to 8000_H, CPUx is woken and continues execution at the instruction following the instruction that was executed before entering the Idle or Sleep Mode.

Note: Before switching into a non-running power-management mode, software should perform a Watchdog service sequence. At the Modify Access, the Watchdog reload value, WDTxCON0.REL, should be programmed such that the wake-up occurs after a period which best meets application requirements. The maximum period between two CPU wake-ups is one-half of the maximum WDT period.

Suspend Mode Support

During a debug session the Watchdog functionality might lead to unintended alarms.

By default the WDTs are disabled when OCDS is enabled. However, it is possible to enable the WDTs also when debug is enabled. Please refer to the manual of your debugger tool for details.

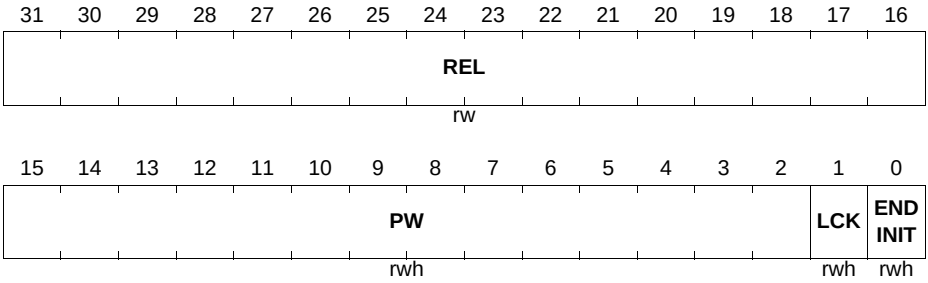
8.4.4.5 Watchdog Timer Registers

Watchdog Timer Control Register 0

This register is written with check data in Check Accesses and Password Accesses. It also stores the timer reload value, password update and the corresponding End-of-Initialization (ENDINIT) control bit during a Modify Access.

There is a WDTxCON0 register for each Watchdog x.

WDTSCON0
Safety WDT Control Register 0 ($0F0_H$) Reset Value: $FFFC\ 000E_H$
WDTCPU0CON0
CPU0 WDT Control Register 0 (100_H) Reset Value: $FFFC\ 000E_H$



Field	Bits	Type	Description
ENDINIT	0	rwh	**End-of-Initialization Control Bit** 0_B Access to Endinit-protected registers is permitted. 1_B Access to Endinit-protected registers is not permitted. This bit must be written with a ‘1’ during a Password Access or Check Access (although this write is only used for the password-protection mechanism and is not stored). This bit must be written with the required ENDINIT update value during a Modify Access.

Field	Bits	Type	Description
LCK	1	rwh	**Lock Bit to Control Access to WDTxCON0** 0_B Register WDTxCON0 is unlocked 1_B Register WDTxCON0 is locked (default after ApplicationReset) The current value of LCK is controlled by hardware. It is cleared after a valid Password Access to WDTxCON0 when WDTxSR.US is 0 (or when WDTxSR.US is 1 and the SMU is in RUN mode), and it is automatically set again after a valid Modify Access to WDTxCON0. During a write to WDTxCON0, the value written to this bit is only used for the password-protection mechanism and is not stored. This bit must be cleared during a Password Access to WDTxCON0, and set during a Modify Access to WDTxCON0. A Check Access does not clear LCK.
PW	[15:2]	rwh	**User-Definable Password Field for Access to WDTxCON0** This bit field is written with an initial password value during a Modify Access. A read from this bitfield returns this initial password, but bits [7:2] are inverted (toggled) to ensure that a simple read/write is not sufficient to service the WDT. If corresponding WDTxSR.PAS = 0 then this bit field must be written with its current contents during a Password Access or Check Access. If corresponding WDTxSR.PAS = 1 then this bit field must be written with the next password in the LFSR sequence during a Password Access or Check Access The default password after Application Reset is 00000000111100_B

Field	Bits	Type	Description
REL	[31:16]	rw	**Reload Value for the WDT (also Time Check Value)** The reload value can be changed during a Modify Access to WDTxCON0 (Default after ApplicationReset is $FFFC_H$).If the Watchdog Timer is enabled and in Normal Timer Mode, it will start counting from this value after a correct Watchdog service. A read from this bitfield always returns the current reload value. During a Password Access or a Check Access this bitfield may be used for additional checks. Writes during such checks have no effect upon the reload value. If corresponding WDTxSR.TCS=0 then this bit field must be written with its current contents during a Password Access or Check Access. If corresponding WDTxSR.TCS=1 then this bit field must be written with an inverted estimate of the current WDTxSR.TIM value during a Password Access or Check Access.

The following register is implemented but is not functional in this product.

WDTCPU1CON0
Reserved Register **($10C_H$)** **Reset Value: $FFFC\ 000F_H$**

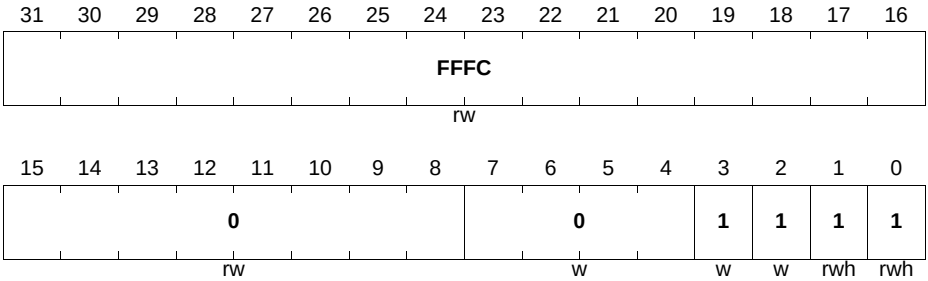
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FFFC															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								0				1	1	1	1
rw								w				w	w	rwh	rwh

Field	Bits	Type	Description
FFFC	[31:16]	rw	**Reserved** Not functional in this device.
0	[15:8]	rw	**Reserved** Not functional in this device.
0	[7:4]	w	**Reserved** Not functional in this device.
1	[3:2]	w	**Reserved** Not functional in this device.
1	[1:0]	rw	**Reserved** Not functional in this device.

The following register is implemented but is not functional in this product.

WDTCPU2CON0
Reserved Register **(118_H)** **Reset Value: FFFC 000F$_H$**



Field	Bits	Type	Description
FFFC	[31:16]	rw	**Reserved** Not functional in this device.
0	[15:8]	rw	**Reserved** Not functional in this device.
0	[7:4]	w	**Reserved** Not functional in this device.
1	[3:2]	w	**Reserved** Not functional in this device.

Field	Bits	Type	Description
1	[1:0]	rw	**Reserved** Not functional in this device.

Watchdog Timer Control Register 1

The register WDTxCON1 for each Watchdog manages operation of the WDT. It includes the disable request, password configuration and frequency selection bits. Each WDTxCON1 register is protected by the corresponding ENDINIT which it controls.

WDTSCON1 has an additional bit CLRIRF which can be used to clear the internal reset status used to detect double SMU (eg WDT) resets.

WDTSCON1
Safety WDT Control Register 1 ($0F4_H$) Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCTR							**TCR**	**PAR**	**UR**	**IR1**	**0**	**DR**	**IR0**	**0**	**CLRIRF**
rw							rw	rw	rw	rw	r	rw	rw	r	rwh

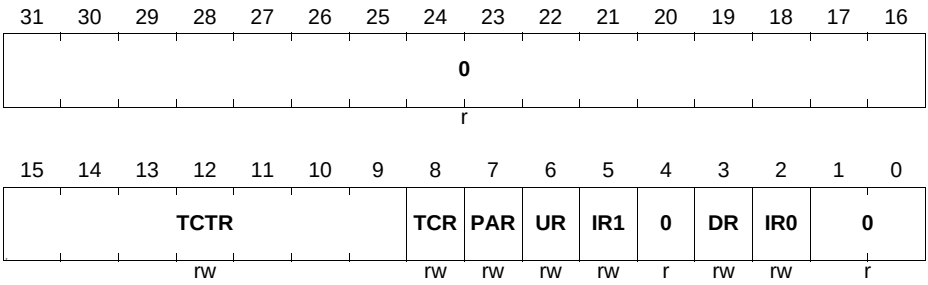
Field	Bits	Type	Description
CLRIRF	0	rwh	**Clear Internal Reset Flag** This bit is used to request a clear of the internal flag which indicates whether a previous SMU reset has already been requested 0_B No action 1_B Request to clear the internal previous-SMU-reset flag This bit can only be modified if WDTSCON0.ENDINIT is cleared. The internal flag is cleared when ENDINIT is set again. As long as ENDINIT is cleared, the internal flag is unchanged and continues to determine the response to a further SMU reset request. When ENDINIT is set again with a valid Modify Access, the internal flag is cleared together with this bit.

Field	Bits	Type	Description
IR1, IR0	5,2	rw	**Input Frequency Request Control** 00_B Request to set input frequency to f_{SPB}/16384. 01_B Request to set input frequency to f_{SPB}/256. 10_B Request to set input frequency to f_{SPB}/64. 11_B Reserved. Do not use. Bit IR0 and IR1 should be programmed together to determine the WDT timer frequency. These bits can only be modified if the corresponding WDTSCON0.ENDINIT is cleared. WDTSSR.ISx are updated by these bit only when ENDINIT is set again. As long as ENDINIT is cleared, WDTSSR.ISx controls the current input frequency of the Watchdog Timer. When ENDINIT is set again, WDTSSR.ISx is updated with the values of IRx.
DR	3	rw	**Disable Request Control Bit** 0_B Request to enable the WDT 1_B Request to disable the WDT This bit can only be modified if the corresponding WDTxCON0.ENDINIT is cleared. WDTSSR.DS is updated when ENDINIT is set again. As long as the ENDINIT is cleared, bit WDTSSR.DS controls the current enable/disable status of the WDT. When the ENDINIT is set again with a Valid Modify Access, WDTSSR.DS is updated with the state of DR.
UR	6	rw	**Unlock Restriction Request Control Bit** 0_B Request to disable SMU restriction of WDT unlock 1_B Request to enable SMU restriction of WDT unlock This bit can only be modified if the corresponding WDTSCON0.ENDINIT is cleared. WDTxSR.US is updated when ENDINIT is set again. As long as the ENDINIT is cleared, bit WDTSSR.US controls whether unlocking is possible at all times or only when the SMU is in the RUN state. When the ENDINIT is set again with a Valid Modify Access, WDTSSR.US is updated with the state of UR.

Field	Bits	Type	Description
PAR	7	rw	**Password Auto-sequence Request Bit** 0_B Request no automatic change of password after each Modify Access or Check Access 1_B Request automatic sequence of password after each Modify Access or Check Access This bit can only be modified if the corresponding WDTxCON0.ENDINIT is cleared. WDTxSR.PAS is updated when ENDINIT is set again. As long as the ENDINIT is cleared, bit WDTxSR.PAS controls password sequencing. When the ENDINIT is set again with a Valid Modify Access, WDTxSR.PAS is updated with the state of PAR.
TCR	8	rw	**Counter Check Request Bit** 0_B Request to check only that REL field is written with existing REL value during Modify Access or Check Access 1_B Request to check that REL field is written with correct TIM Count (within tolerance of WDTxSR.TCT) during Modify Access or Check Access This bit can only be modified if the corresponding WDTxCON0.ENDINIT is cleared. WDTxSR.TCS is updated when ENDINIT is set again. As long as the ENDINIT is cleared, bit WDTxSR.TCS controls whether counter check is enabled. When the ENDINIT is set again with a Valid Modify Access, WDTxSR.TCS is updated with the state of TCR
TCTR	[15:9]	rw	**Timer Check Tolerance Request** This bit can only be modified if the corresponding WDTxCON0.ENDINIT is cleared. WDTxSR.TCT is updated when ENDINIT is set again. As long as the ENDINIT is cleared, bit WDTxSR.TCT controls the tolerance of timer checks. When the ENDINIT is set again with a Valid Modify Access, WDTxSR.TCT is updated with the state of TCTR
0	1,4, [31:16]	r	**Reserved** Read as 0; should be written with 0.

The register WDTxCON1 for each CPU Watchdog manages operation of the WDT. It includes the disable request, password configuration and frequency selection bits. Each WDTxCON1 register is protected by the corresponding ENDINIT which it controls.

WDTCPU0CON1
CPU0 WDT Control Register 1 (104_H) Reset Value: 0000 0000_H



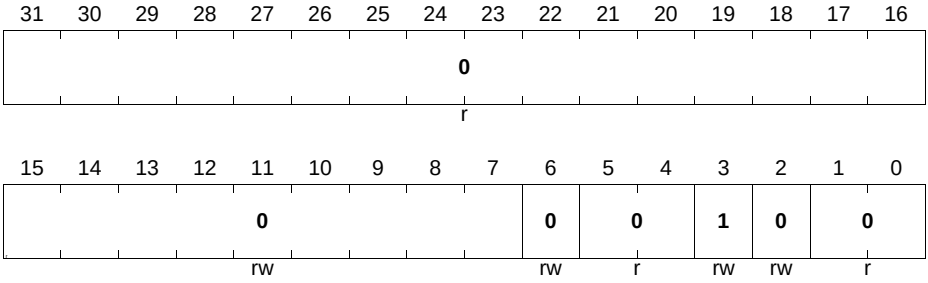
Field	Bits	Type	Description
IR1, IR0	5,2	rw	**Input Frequency Request Control** 00_B Request to set input frequency to f_{SPB}/16384. 01_B Request to set input frequency to f_{SPB}/256. 10_B Request to set input frequency to f_{SPB}/64. 11_B Reserved. Do not use. Bit IR0 and IR1 should be programmed together to determine the WDT timer frequency. These bits can only be modified if the corresponding WDTSCON0.ENDINIT is cleared. WDTSSR.ISx are updated by these bit only when ENDINIT is set again. As long as ENDINIT is cleared, WDTSSR.ISx controls the current input frequency of the Watchdog Timer. When ENDINIT is set again, WDTSSR.ISx is updated with the values of IRx.

Field	Bits	Type	Description
DR	3	rw	**Disable Request Control Bit** 0_B Request to enable the WDT 1_B Request to disable the WDT This bit can only be modified if the corresponding WDTxCON0.ENDINIT is cleared. WDTxSR.DS is updated when ENDINIT is set again. As long as the ENDINIT is cleared, bit WDTxSR.DS controls the current enable/disable status of the WDT. When the ENDINIT is set again with a Valid Modify Access, WDTxSR.DS is updated with the state of DR.
UR	6	rw	**Unlock Restriction Request Control Bit** 0_B Request to disable SMU restriction of WDT unlock 1_B Request to enable SMU restriction of WDT unlock This bit can only be modified if the corresponding WDTxCON0.ENDINIT is cleared. WDTxSR.US is updated when ENDINIT is set again. As long as the ENDINIT is cleared, bit WDTxSR.US controls whether unlocking is possible at all times or only when the SMU is in the RUN state. When the ENDINIT is set again with a Valid Modify Access, WDTxSR.US is updated with the state of UR.
PAR	7	rw	**Password Auto-sequence Request Bit** 0_B Request no automatic change of password after each Modify Access or Check Access 1_B Request automatic sequence of password after each Modify Access or Check Access This bit can only be modified if the corresponding WDTxCON0.ENDINIT is cleared. WDTxSR.PAS is updated when ENDINIT is set again. As long as the ENDINIT is cleared, bit WDTxSR.PAS controls password sequencing. When the ENDINIT is set again with a Valid Modify Access, WDTxSR.PAS is updated with the state of PAR.

Field	Bits	Type	Description
TCR	8	rw	**Counter Check Request Bit** 0_B Request to check only that REL field is written with existing REL value during Modify Access or Check Access 1_B Request to check that REL field is written with correct TIM Count (within tolerance of WDTxSR.TCT) during Modify Access or Check Access This bit can only be modified if the corresponding WDTxCON0.ENDINIT is cleared. WDTxSR.TCS is updated when ENDINIT is set again. As long as the ENDINIT is cleared, bit WDTxSR.TCS controls whether counter check is enabled. When the ENDINIT is set again with a Valid Modify Access, WDTxSR.TCS is updated with the state of TCR
TCTR	[15:9]	rw	**Timer Check Tolerance Request** This bit can only be modified if the corresponding WDTxCON0.ENDINIT is cleared. WDTxSR.TCT is updated when ENDINIT is set again. As long as the ENDINIT is cleared, bit WDTxSR.TCT controls the tolerance of timer checks. When the ENDINIT is set again with a Valid Modify Access, WDTxSR.TCT is updated with the state of TCTR
0	0,1,4, [31:16]	r	**Reserved** Read as 0; should be written with 0.

The following registers are implemented for reuse but are not functional in the TC21x/TC22x/TC23x.

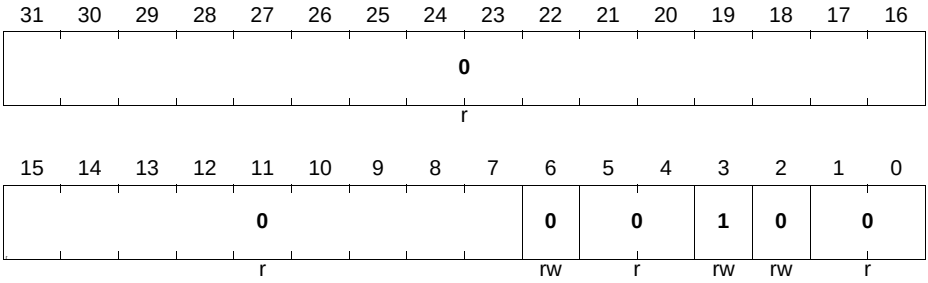
WDTCPU1CON1
Reserved Register **(110_H)** **Reset Value: 0000 0008$_H$**



Field	Bits	Type	Description
0	[31:16],5,4,1,0	r	**Reserved** Not functional in this device.
0	[15:6],2	rw	**Reserved** Not functional in this device.
1	3	rw	**Reserved** Not functional in this device.

The following registers are implemented for reuse but are not functional in the TC21x/TC22x/TC23x.

WDTCPU2CON1
Reserved Register **($11C_H$)** **Reset Value: 0000 0008$_H$**

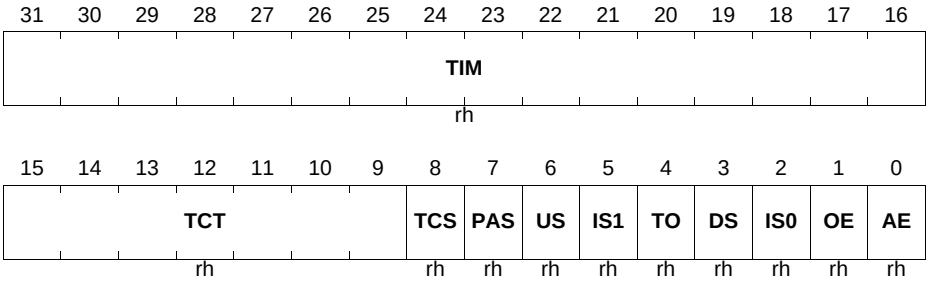


Field	Bits	Type	Description
0	[31:7], 5,4,1,0	r	**Reserved** Not functional in this device.
0	6,2	rw	**Reserved** Not functional in this device.
1	3	rw	**Reserved** Not functional in this device.

Watchdog Timer Status Register

The WDTxSR registers show the current state of each WDT. Status include bits indicating Time-Out, enable/disable status, input clock status, and access error status.

WDTSSR
Safety WDT Status Register **($0F8_H$)** **Reset Value: $FFFC\ 0010_H$**
WDTCPU0SR
CPU0 WDT Status Register **(108_H)** **Reset Value: $FFFC\ 0010_H$**



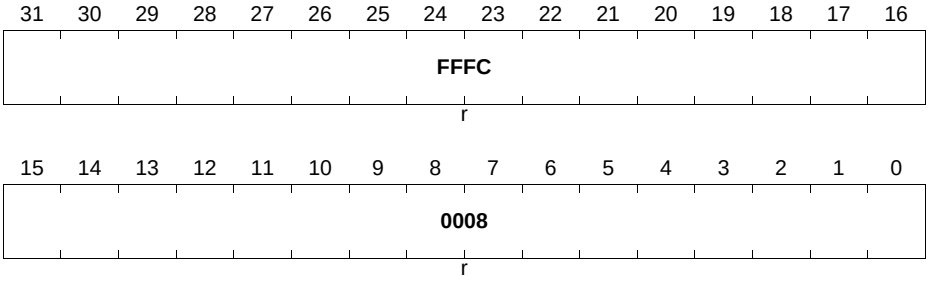
Field	Bits	Type	Description
AE	0	rh	**Watchdog Access Error Status Flag** 0_B No Watchdog access error 1_B A Watchdog access error has occurred This bit is set when an illegal Password Access or Modify Access to register WDTxCON0 was attempted. This bit is only cleared when WDTxCON0.ENDINIT is set during a valid Modify Access

Field	Bits	Type	Description
OE	1	rh	**Watchdog Overflow Error Status Flag** 0_B No Watchdog overflow error 1_B A Watchdog overflow error has occurred This bit is set when the WDT overflows from $FFFF_H$ to 0000_H. This bit is only cleared when WDTxCON0.ENDINIT is set to 1 during a valid Modify Access.
IS1, IS0	5,2	rh	**Watchdog Input Clock Status** 00_B WDT counter frequency is f_{SPB}/16384. 01_B WDT counter frequency is f_{SPB}/256. 10_B WDT counter frequency is f_{SPB}/64. 11_B Reserved. Do not use. Bit IS0 and IS1 should be programmed together. These bits are updated with the state of bits WDTxCON1.IRx after WDTxCON0.ENDINIT is written with 1 during a valid Modify Access to register WDTxCON0.
DS	3	rh	**Watchdog Enable/Disable Status Flag** 0_B WDT is enabled (default after ApplicationReset) 1_B WDT is disabled This bit is updated with the state of bit WDTxCON1.DR (after WDTxCON0.ENDINIT is set during a Valid Modify Access to register WDTxCON0) and it is cleared when Time-Out mode is entered.
TO	4	rh	**Watchdog Time-Out Mode Flag** 0_B The Watchdog is not operating in Time-Out Mode 1_B The Watchdog is operating in Time-Out Mode (default after ApplicationReset) This bit is set when Time-Out Mode is entered. It is automatically cleared when Time-Out Mode is left.

Field	Bits	Type	Description
US	6	rh	**SMU Unlock Restriction Status Flag** 0_B WDT unlock permitted at any time 1_B WDT unlock only permitted when the SMU is in the RUN state. WDTxCON0.LCK will not be unlocked by a valid Password Access if this bit is '1' and the SMU is not in the RUN state
PAS	7	rh	**Password Auto-sequence Status Flag** 0_B No change of password after each Modify Access or Check Access 1_B Automatically sequence the password after each Modify Access or Check Access This bit is updated with the state of bit WDTxCON1.PAR after WDTxCON0.ENDINIT is written with 1 during a valid Modify Access to register WDTxCON0.
TCS	8	rh	**Timer Check Status Flag** 0_B Check only that REL field is written with existing REL value during Modify Access or Check Access 1_B Check that REL field is written with inverted estimated TIM value (within +/- TCT value) during Password Access or Check Access This bit is updated with the state of bit WDTxCON1.TCR after WDTxCON0.ENDINIT is written with 1 during a Valid Modify Access to register WDTxCON0.
TCT	[15:9]	rh	**Timer Check Tolerance** This field determines the tolerance of the timer check during Password or Check Access (See TCS).This bit is updated with the state of bit WDTxCON1.TCTR after WDTxCON0.ENDINIT is written with 1 during a Valid Modify Access to register WDTxCON0.
TIM	[31:16]	rh	**Timer Value** Reflects the current content of the WDT.

The following registers are implemented for reuse but are not functional in the TC21x/TC22x/TC23x.

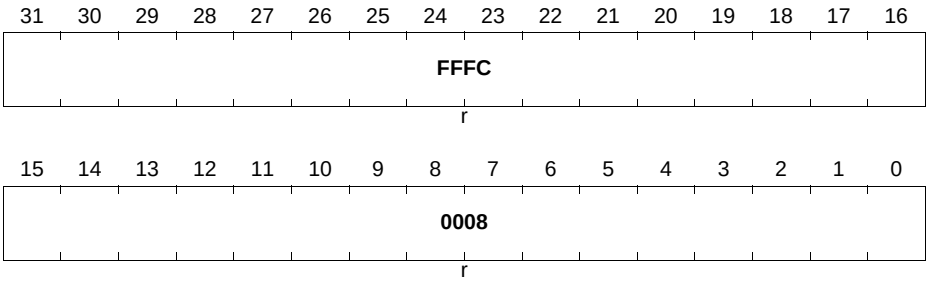
WDTCPU1SR
CPU1 WDT Status Register **(114_H)** **Reset Value: FFFC 0008$_H$**



Field	Bits	Type	Description
FFFC	[31:16]	r	**Reserved** Not functional in this device
0008	[15:0]	r	**Reserved** Not functional in this device

The following registers are implemented for reuse but are not functional in the TC21x/TC22x/TC23x.

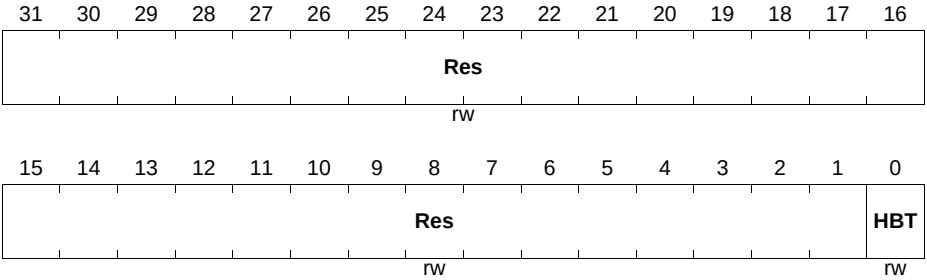
WDTCPU2SR
CPU2 WDT Status Register **(120_H)** **Reset Value: FFFC 0008$_H$**



Field	Bits	Type	Description
FFFC	[31:16]	r	**Reserved** Not functional in this device
0008	[15:0]	r	**Reserved** Not functional in this device

Safety Heartbeat Register

SAFECON
Safety Heartbeat Register **(150_H)** **Reset Value: XXXX XXXX$_H$**



Field	Bits	Type	Description
HBT	0	rw	**Heartbeat** This bit can be used by software to provide a “heartbeat” indication to an external safety monitoring device. It directly controls the output on the heartbeat ports. WDTxLCK P20.9, P20.8, P20.7 and/or P20.6 (Subject to port configuration). 0_B Port drives low (reset value) 1_B Port drives high Writes are only possible when Safety ENDINIT=0
Res	[31:1]	rw	**Reserved** Content must not be changed. Write back only the original read value.

8.4.5 Emergency Stop Output Control

The emergency stop feature of the TC21x/TC22x/TC23x provides a fast reaction to an emergency event without the intervention of software. In response to the emergency event, selected output ports can be immediately placed into a defined state (for more information see the port chapter).

An emergency stop may be triggered by either of the following emergency events:

- A transition on the port which is configured as the Emergency Stop input
- An SMU alarm event or SMU command which is enabled and configured to generate a port emergency stop (PES). See SMU Chapter for details.

Figure 8-42 shows a diagram of the emergency stop input logic. This logic is controlled by the SCU Emergency Stop Register EMSR.

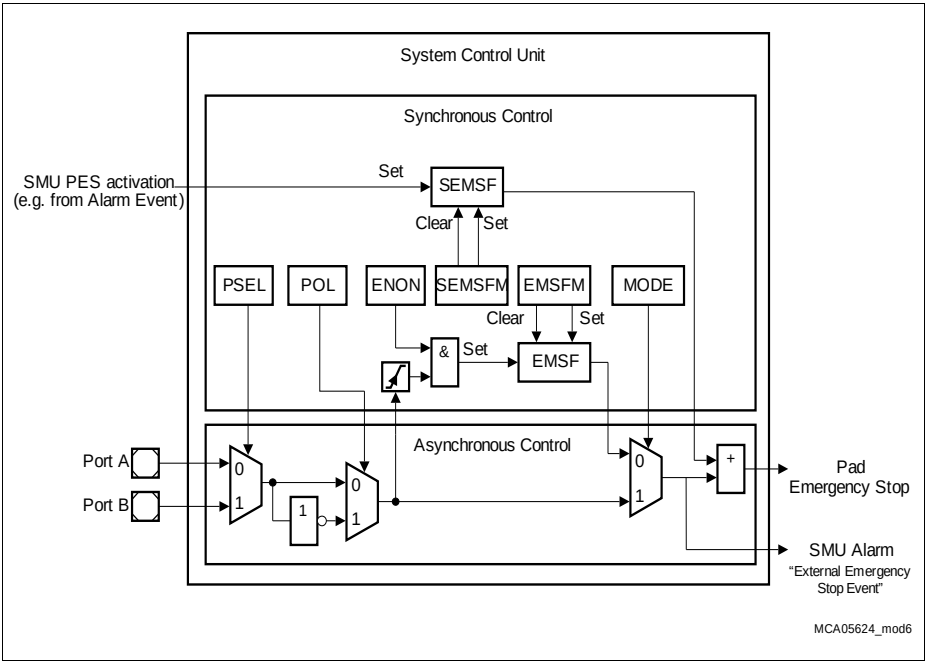


Figure 8-42 Emergency Stop Control

8.4.5.1 Port Triggered Emergency Stop

This can be configured to trigger on either transition edge of either one of two ports.

The two input port options for TC21x/TC22x/TC23x are (A) P 33.8 and (B) P21.2.

The emergency stop control logic for the ports can basically operate in two modes:

- Synchronous Mode (default after reset):
  Emergency case is activated by hardware and released by software.
- Asynchronous Mode:
  Emergency case is activated and released by hardware.

In Synchronous Mode (selected by EMSR.MODE = 0), the port signal is sampled for a inactive-to-active level transition, and an emergency stop flag EMSR.EMSF is set if the transition is detected. The setting of EMSR.EMSF activates the emergency stop. A port triggered emergency state can only be terminated by clearing EMSR.EMSF via software (Write EMSR.EMSFM with 10_B). The synchronous control logic is clocked by the system bus clock f_{SPB}. This results in a small delay between the port signal and emergency stop signal generation.

In Asynchronous Mode (selected by EMSR.MODE = 1), the occurrence of an active level at the port input immediately activates the emergency stop signal. Of course, a valid-to-invalid transition of the port input (emergency case is released) also immediately deactivates the emergency stop signal.

The EMSR.POL bit determines the active level of the input signal from the port. The EMSR.MODE bit selects Synchronous or Asynchronous Mode for emergency stop signal generation and the EMSR.PSEL bit selects which of the two ports is used as the emergency stop trigger.

8.4.5.2 SMU Event Triggered Emergency Stop

The Safety Alarm(s) which can trigger an Emergency Stop are configured and enabled within the Safety Management Unit (SMU). All SMU triggered Emergency Stop cases are in Synchronous Mode, regardless of the state of EMSR.MODE. The safety emergency stop flag EMSR.SEMSF is set when a configured and enabled SMU Safety Alarm occurs. The setting of EMSR.SEMSF activates the emergency stop. An SMU triggered emergency state can only be terminated by clearing the EMSR.SEMSF via software (Write EMSR.SEMSFM with 10_B). The synchronous control logic is clocked by the system bus clock f_{SPB}.

8.4.5.3 Emergency Stop Register

The Emergency Stop Register EMSR contains control and status bits/flags of the emergency stop input logic.

EMSR
Emergency Stop Register **(0FC$_H$)** **Reset Value: 0000 0001$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				SEMSFM		EMSFM		0						SEMSF	EMSF
r				w		w		r						rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												PSEL	ENON	MODE	POL
r												wr	rw	rw	rw

Field	Bits	Type	Description
POL	0	rw	**Input Polarity** This bit determines the polarity of the configured Emergency Stop input. 0_B Input is active high 1_B Input is active low
MODE	1	rw	**Mode Selection** This bit determines the operating mode of the emergency stop signal. 0_B Synchronous Mode selected; emergency stop is derived from the state of flag EMSF 1_B Asynchronous Mode selected; emergency stop is directly derived from the state of the input signal
ENON	2	rw	**Enable ON** This bit enables the setting of flag EMSF by an inactive-to-active level transition of input signal. 0_B Setting of EMSF is disabled 1_B Setting of EMSF is enabled

Field	Bits	Type	Description
PSEL	3	rw	**PORT Select** This bit selects which one of the two Emergency Stop port options is monitored. 0_B Port A is used as Emergency Stop input 1_B Port B is used as Emergency Stop input
0	[15:4]	r	**Reserved** Read as 0; should be written with 0.
EMSF	16	rh	**Emergency Stop Flag** This bit indicates that a synchronous mode port-triggered emergency stop condition has occurred. 0_B An emergency stop has not occurred 1_B An emergency stop has occurred and emergency stop state becomes active (if MODE = 0)
SEMSF	17	rh	**SMU Emergency Stop Flag** This bit indicates that an SMU Safety Alarm triggered emergency stop condition has occurred. 0_B An emergency stop has not occurred 1_B An emergency stop has occurred and emergency stop state becomes active
0	[23:18]	r	**Reserved** Read as 0; should be written with 0.
EMSFM	[25:24]	w	**Emergency Stop Flag Modification** This bit field sets or clears flag EMSF via software. 00_B EMSF remains unchanged 01_B EMSF becomes set 10_B EMSF becomes cleared 11_B EMSF remains unchanged EMSFM is always read as 00_B.
SEMSFM	[27:26]	w	**SMU Emergency Stop Flag Modification** This bit field sets or clears flag SEMSF via software. 00_B SEMSF remains unchanged 01_B SEMSF becomes set 10_B SEMSF becomes cleared 11_B SEMSF remains unchanged SEMSFM is always read as 00_B.
0	[31:28]	r	**Reserved** Read as 0; should be written with 0.

8.4.6 LBIST Support

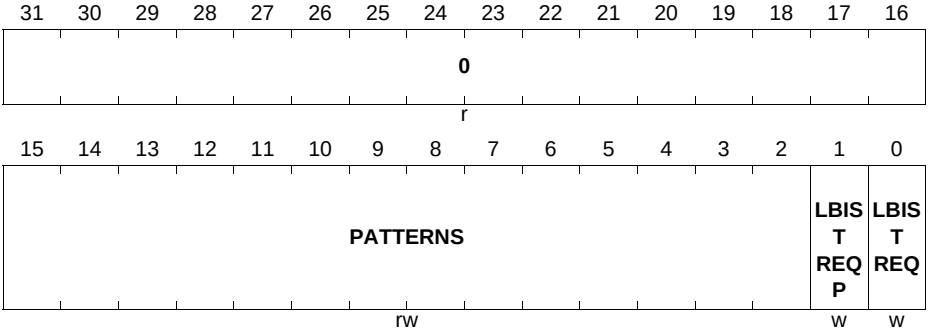
For TC21x/TC22x/TC23x an automatic executed scan-test mechanism is implemented, allowing the structural verification of the silicon in an application system. This process is controlled and monitored by the LBIST registers.

8.4.6.1 LBIST Control Register

The LBISTCTRL Control Register provides the link between software and the LBIST-controller.

LBISTCTRL0 - Logic BIST Control 0 Register

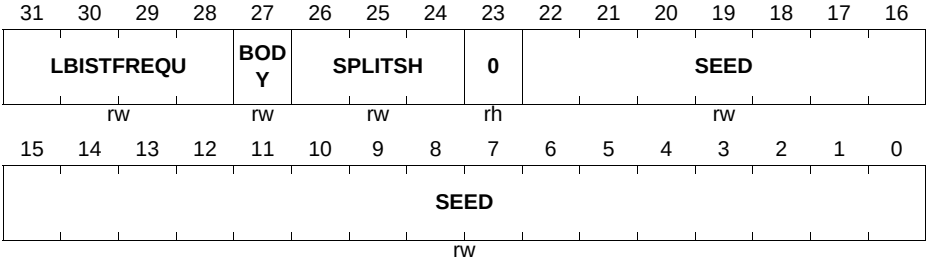
LBISTCTRL0
Logic BIST Control 0 Register **(164_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
LBISTREQ	0	w	**LBIST Request** If written high this bit requests the execution of an automatic scan-test procedure. Whether the request is successful or not depends on whether the LBIST procedure has already been executed since the last power-on-reset. If read this bit always returns a '0'. *Note: LBIST execution time depends on the number of scan-loads as defined in the PATTERNS field.*
LBISTREQP	1	w	**LBIST Request Protection Bit** Protect LBISTREQ against unintended changes. 0_B LBISTREQ is not changed. 1_B LBISTREQ is updated by this write access.
PATTERNS	[15:2]	rw	**LBIST Pattern Number** This field defines the number of scan-patterns (i.e. scan-loads), which will be executed during the LBIST-procedure. The two LSBs of scan pattern number are always assumed as '1' so LBIST patterns number can only be controlled with an accuracy of 4.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

LBISTCTRL1 - Logic BIST Control 1 Register

LBISTCTRL1
Logic BIST Control 1 Register **(168_H)** **Reset Value: 0000 0000$_H$**

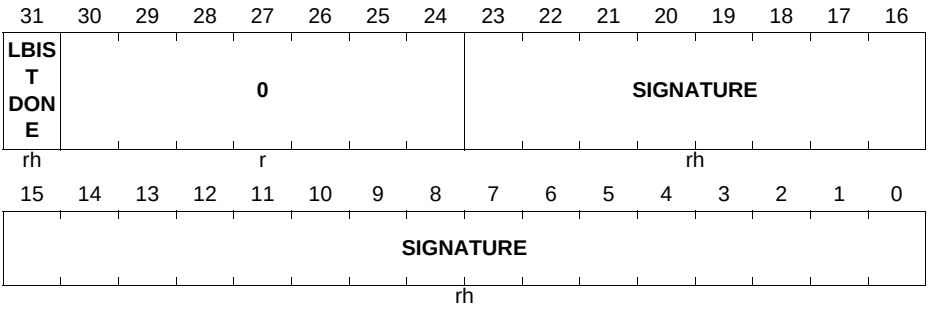


Field	Bits	Type	Description
SEED	[22:0]	rw	**LBIST Seed** This field determines which pattern is applied during LBIST execution.
0	23	r	**Reserved** Read as 0; should be written with 0.
SPLITSH	[26:24]	rw	**LBIST Split-Shift Selection** The value of this bit will allow to run LBIST with partitioned scan-shift operation in order to reduce the power consumption. $0xx_B$, Concurrent scan-shift is selected. $1x0_B$, Partitioned scan-shift is selected (four scan partitions). $1x1_B$, Partitioned scan-shift is selected (two scan partitions).
BODY	27	rw	**Body Application Indicator** The value of this bit will determine the static reset behavior of all GPIOs during LBIST execution. If set to low GPIOs will show a weak pull-up behavior, if set to high GPIOs are constrained to tri-state. A high value must be written to this bit in case LBIST shall be executed for body applications.

Field	Bits	Type	Description
LBISTFREQU	[31:28]	rw	**LBIST Frequency Selection** Through this register-field a pre-scaler factor between 1..16 is selectable for LBIST operation clock (derived from EVR-oscillator). This will allow to determine the LBIST scan-shift frequency. Value of these bits will be mirrored inside of LBIST-controller and become effective if a new LBIST-procedure has been successfully initiated via LBISTCTRL0.LBISTREQ.

LBISTCTRL2 - Logic BIST Control 1 Register

LBISTCTRL2
Logic BIST Control 2 Register **($16C_H$)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
SIGNATURE	[23:0]	rh	**LBIST Signature** This field reflects the MISR signature from the last LBIST execution. It is mirrored from LBIST-controller and is only valid if LBISTCTRL0.LBISTDONE = 1.
0	[30:24]	r	**Reserved** Read as 0; should be written with 0.

Field	Bits	Type	Description
LBISTDONE	31	rh	**LBIST Execution Indicator** This bit indicates if since the last power-on-reset an automatic scan-test procedure has been executed: 0_B No LBIST was executed since last power-on-reset. 1_B LBIST was executed since last power-on-reset.

8.4.7 Global Overlay Controls

The following registers control the Global Overlay functionality.

8.4.7.1 Global Overlay Control

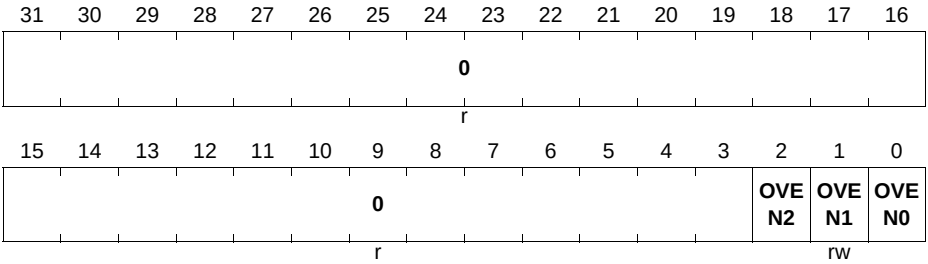
Two registers globally control the overlay operation for all CPUs:

- Overlay Enable Register OVCENABLE disables or enables data access overlay individually for each CPU.
- Overlay Control Register OVCCON can be used to perform the following actions on a selected set of CPUs:
  – concurrently enable / disable selected overlay blocks,
  – concurrently disable overlay blocks,
  – invalidate data cache.

All overlay control registers are reset to their default values with the Application Reset . A special debug reset is not considered.

The external overlay feature is not available in product variants offering ADAS functionality.

OVCENABLE
Overlay Enable Register **(1E0H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
OVEN0	0	rw	**Overlay Enable 0** 0_B OVC is disabled on CPU0. All Overlay redirections are disabled regardless of the state of OVC0_RABRy.OVEN. 1_B OVC is enabled on CPU0.
OVEN1	1	rw	**Reserved in this Product**
OVEN2	2	rw	**Reserved in this Product**
0	[31:3]	r	**Reserved** Read/write 0.

OVCCON
Overlay Control Register **(1E4H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						POVCONF	OVCONF	0					DCINVAL	OVSTP	OVSTRT
r						w	rw	r					w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													CSEL2	CSEL1	CSEL0
r													w	w	w

Field	Bits	Type	Description
CSEL0	0	w	**CPU Select 0** 0_B CPU0 not affected, 1_B Action selected by OVSTRT, OVSTP, DCINVAL bits, set by the same register write access, is applied to CPU0. Return 0 if read.
CSEL1	1	r	**Reserved in this Product** Return 0 if read.
CSEL2	2	r	**Reserved in this Product** Return 0 if read.
0	[15:3]	r	**Reserved** Read/write 0.
OVSTRT	16	w	**Overlay Start** 0_B No action 1_B For each CPU selected with CSEL, all the blocks selected with OVCx_OSEL will be activated. In the selected CPUs all the blocks deselected with OVCx_OSEL will be deactivated. CPUs which are not selected are not affected. No action is taken if OVSTP is also set. Return 0 if read.

Field	Bits	Type	Description
OVSTP	17	w	**Overlay Stop** 0_B No action 1_B For CPUs selected with CSEL, all the overlay blocks are deactivated. OVCx_RABRy.OVEN bits are cleared. CPUs which are not selected are not affected No action is taken if OVSTRT is also set. Return 0 if read.
DCINVAL	18	w	**Data Cache Invalidate** No function in devices without data cache in CPU. 0_B No action 1_B Data Cache Lines in DMI are invalidated[1)] Data Cache is affected only in the CPUs selected with CSEL. Return 0 if read.
0	[23:19]	r	**Reserved** Read/write 0.
OVCONF	24	rw	**Overlay Configured** Overlay configured status bit 0_B Overlay is not configured or it has been already started 1_B Overlay block control registers are configured and ready for overlay start This bit may be used as handshake bit between a debug device (via JTAG interface and Cerberus) and CPU(s).
POVCONF	25	w	**Write Protection for OVCONF** 0_B OVCONF remains unchanged. 1_B OVCONF can be changed with write access to register OVCCON This bit enables OVCONF write during OVCCON write. Return 0 if read.
0	[31:26]	r	**Reserved** Read/write 0.

1) Dirty (modified) cache lines are not effected by this operation. If data cache contains modified data, it is not invalidated, and has to be written-back and invalidated by the user. Therefore, it is highly recommended to either: access overlaid data in read-only mode, or use only non-cached access.

.

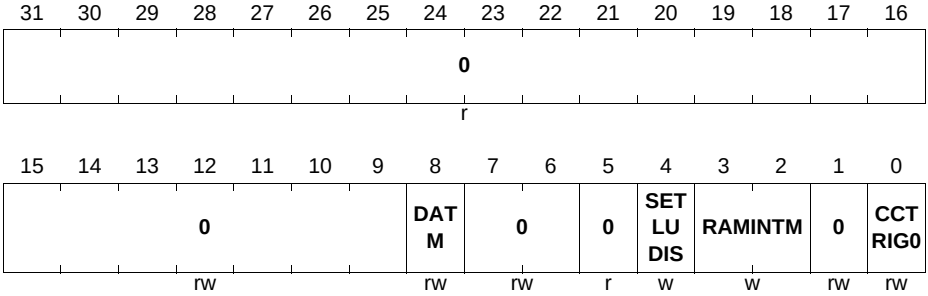
8.4.8 Miscellaneous System Control

This section collects different registers that serve various system purposes.

8.4.8.1 System Control Register

This register controls various functionality used by the SCU but that are located outside of the module. Additionally some functions for other modules are included.

SYSCON
System Control Register **($07C_H$)** **Reset Value: 0000 0000$_H$**



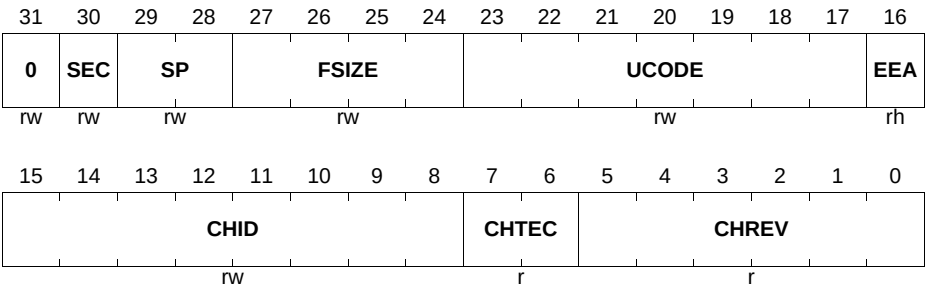
Field	Bits	Type	Description
CCTRIG0	0	rw	**Capture Compare Trigger 0** This bit is used to trigger the Synchronous Start feature of the CCU6.
RAMINTM	[3:2]	w	**RAM Integrity Modify** 00_B No effect 01_B Set STSTAT.RAMINT (No effect in test mode) 11_B No effect 10_B Clear STSTAT.RAMINT
SETLUDIS	4	w	**Set Latch Update Disable** Setting this bit sets bit STSTAT.LUDIS. Clearing this bit has no effect. This bit reads always as zero.
DATM	8	rw	**Disable Application Test Mode (ATM)** 0_B ATM not disabled 1_B ATM disabled

Field	Bits	Type	Description
0	[15:9],7, 6,1	rw	**Reserved** Write only the read value
0	5	r	**Reserved** Read as 0
0	[31:16]	r	**Reserved** Read as 0

8.4.8.2 Identification Registers

The CHIPID register allows the user to determine the product, package option, FSI microcode version and chip revision (silicon step).

CHIPID
Chip Identification Register **(140_H)** **Reset Value: XXXX XXXX$_H$**

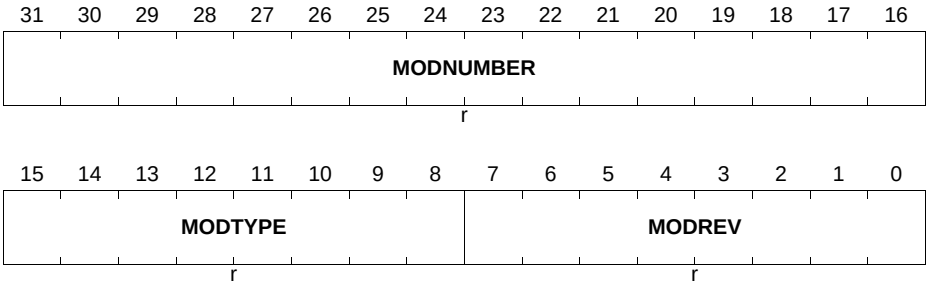


Field	Bits	Type	Description
CHREV	[5:0]	r	**Chip Revision Number** This bit field indicates the revision number of the TC21x/TC22x/TC23x device. The value of this bit field is defined in the TC21x/TC22x/TC23x Data Sheet. Bits [3:0] are used to indicate the steps. These updates can be done with any metal-fix or FW ROM change. Bits [5:4] define the major silicon design steps (A, B, C, D). These bits can be changed only with a major redesign. Note that this specification refers to the latest available productive design step. Earlier silicon functionality may vary. Consult the appropriate delta documentation for details. $00XXXX_B$ A-step silicon $01XXXX_B$ B-step silicon $10XXXX_B$ C-step silicon $11XXXX_B$ D-step silicon

Field	Bits	Type	Description
CHTEC	[7:6]	r	**Chip Family** These bits indicate the product family and are changed only with a redesign. 00_B Reserved 01_B AURIX Gen1 Family 10_B Reserved 11_B Reserved
CHID	[15:8]	rw	**Chip Identification Number** This bit field defines the product by a unique number. This number may be used by SW to identify the system architecture (e.g. number of available CPUs). See Product Datasheet Addendum for more details.
EEA	16	rh	**Emulation Extension Available** Indicates if the emulation extension is available or not. 0_B EEC is not available 1_B EEC is available
UCODE	[23:17]	rw	**µCode Version** This bit field displays the Version X.Y of the flash µCode.
FSIZE	[27:24]	rw	**Program Flash Size** This bit field indicates available program flash size for this device. Detailed information is shown in the Data Sheet. 0000_B256 KByte Program Flash (TC2xxx-4Fx) 0001_B0.5 MByte Program Flash (TC2xxx-8Fx) 0010_B1.0 MByte Program Flash (TC2xxx-16Fx) 0011_B1.5 MByte Program Flash (TC2xxx-24Fx) 0100_B2.0 MByte Program Flash (TC2xxx-32Fx) 0101_B2.5 MByte Program Flash (TC2xxx-40Fx) 0110_B3.0 MByte Program Flash (TC2xxx-48Fx) 0111_B4.0 MByte Program Flash (TC2xxx-64Fx) 1000_B5.0 MByte Program Flash (TC2xxx-80Fx) 1001_B6.0 MByte Program Flash (TC2xxx-96Fx) 1010_B7.0 MByte Program Flash (TC2xxx-112Fx) 1011_B8.0 MByte Program Flash (TC2xxx-128Fx) 1100_BReserved, do not use this combination 1101_BReserved, do not use this combination 1110_BReserved, do not use this combination 1111_BReserved, do not use this combination

Field	Bits	Type	Description
SP	[29:28]	rw	**Speed** This bit field indicates the maximum allowed frequency for the application CPU(s) 00_B Default speed grade 01_B Speed grade two 10_B Speed grade three 11_B Speed grade four
SEC	30	rw	**Security Device** This bit field indicates whether the product has a Hardware Security Module (HSM) 0_B No Hardware Security Module 1_B Hardware Security Module is available
0	31	rw	**Reserved** Must not be written.

ID
Identification Register **(008_H)** **Reset Value: $00C4\ C081_H$**

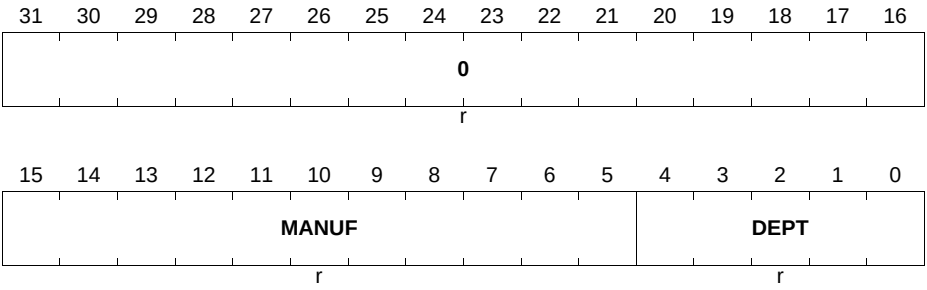


Field	Bits	Type	Description
MODREV	[7:0]	r	**Module Revision Number** This bit field indicates the revision number of the AURIX SCU module (81_H).
MODTYPE	[15:8]	r	**Module Type** This bit field is $C0_H$. It defines a 32-bit module

Field	Bits	Type	Description
MODNUMBER	[31:16]	r	**Module Number** This bit field defines the module identification number. The idenfication number for the AURIX SCU is $00C4_H$

MANID

Manufacturer Identification Register (144_H) **Reset Value: $0000\ 1820_H$**



Field	Bits	Type	Description
DEPT	[4:0]	r	**Department Identification Number** = 00_H: indicates the ATV Microcontroller department within Infineon Technologies.
MANUF	[15:5]	r	**Manufacturer Identification Number** This is a JEDEC normalized manufacturer code. MANUF = $C1_H$ stands for Infineon Technologies.
0	[31:16]	r	**Reserved** Read as 0.

8.4.8.3 SCU Access Restriction Registers

The Access Enable Register 0 restricts write access to all SCU registers so that they may only be written by specified bus masters (eg CPUs). See the Bus chapter for the mapping of TAG ID to specific system masters and CPUs).

ACCEN0
Access Enable Register 0 **($3FC_H$)** **Reset Value: $FFFF\ FFFF_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN 31	EN 30	EN 29	EN 28	EN 27	EN 26	EN 25	EN 24	EN 23	EN 22	EN 21	EN 20	EN 19	EN 18	EN 17	EN 16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN 15	EN 14	EN 13	EN 12	EN 11	EN 10	EN9	EN8	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n = 0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the SCU kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

ACCEN1
Access Enable Register 1 **($3F8_H$)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
0	[31:0]	r	**Reserved** Read as 0; should be written with 0.

8.4.9 SCU Register Address

Table 8-28 Registers Address Spaces - SCU Kernel Registers

Module	Base Address	End Address	Note
SCU	$F003\ 6000_H$	$F003\ 63FF_H$	-

8.4.10 SCU Kernel Registers

This section describes the kernel registers of the SCU module. Most of SCU kernel register names described in this section will be referenced in other parts of the TC21x/TC22x/TC23x User's Manual by the module name prefix "SCU_".

SCU Kernel Register Overview

Table 8-29 Register Overview of SCU (Offset from Main Register Base)

Short Name	Long Name	Offset Addr. [1]	Access Mode		Reset	Description See
			Read	Write		
–	Reserved	000_H - $00C_H$	BE	BE	–	–
SYSCON	System Control Register	$07C_H$	U, SV	U, SV, P	System Reset	Page 8-271
–	Reserved	090_H - 098_H	BE	BE	–	–
DTSSTAT	Die Temperature Sensor Status Register	$0E0_H$	U, SV	BE	Application Reset	Page 8-223
DTSCON	Die Temperature Sensor Control Register	$0E4_H$	U, SV	U, SV, P	Application Reset	Page 8-222
WDTSCON0	Safety WDT Control Register 0	$0F0_H$	U, SV	U, SV, 32, P	Application Reset	Page 8-241
WDTSCON1	Safety WDT Control Register 1	$0F4_H$	U, SV	SV, SE, P	Application Reset	Page 8-245
WDTSSR	Safety WDT Status Register	$0F8_H$	U, SV	BE	Application Reset	Page 8-252

Table 8-29 Register Overview of SCU (Offset from Main Register Base)

Short Name	Long Name	Offset Addr. 1)	Access Mode		Reset	Description See
			Read	Write		
EMSR	Emergency Stop Register	$0FC_H$	U, SV	SV, SE, P	Application Reset	Page 8-259
WDTCPU0CON0	CPU0 WDT Control Register 0	100_H	U, SV	U, SV, 32,(CPU $0^{2)}$)	Application Reset	Page 8-241
WDTCPU0CON1	CPU0 WDT Control Register 1	104_H	U, SV	SV, CE0, P	Application Reset	Page 8-248
WDTCPU0SR	CPU0 WDT Status Register	108_H	U, SV	BE	Application Reset	Page 8-252
–	Reserved	$10C_H$-114_H	U, SV	–	–	
–	Reserved	118_H-120_H	U, SV	–	–	
LCLCON0	CPU0 Lockstep Control Register	134_H	U, SV	BE	Cold Power-On Reset	Page 8-217
LCLCON1	CPU1 Lockstep Control Register	138_H	U, SV	BE	Cold Power-On Reset	Page 8-218
LCLTEST	Lockstep Test Register	$13C_H$	U,SV	U,SV, P	System Reset	Page 8-219
CHIPID	Chip Identification Register	140_H	U, SV	BE	System Reset	Page 8-273
MANID	Manufacture Identification Register	144_H	U, SV	BE	System Reset	Page 8-276
–	Reserved	148_H - 160_H	BE	BE	–	–
LBISTCTRL0	Logic BIST Control 0	164_H	U, SV	SV, SE, P	System Reset	Page 8-262
LBISTCTRL1	Logic BIST Control 1	168_H	U, SV	SV, SE, P	System Reset	Page 8-263

Table 8-29 Register Overview of SCU (Offset from Main Register Base)

Short Name	Long Name	Offset Addr. [1]	Access Mode		Reset	Description See
			Read	Write		
LBISTCTRL2	Logic BIST Control 2	$16C_H$	U, SV	BE	System Reset	Page 8-264
–	Reserved	194_H	BE	BE	–	–
–	Reserved	$1DC_H$	BE	BE	–	–
OVCENABLE	Overlay Enable	$1E0_H$	U,SV	SV,SE, P	Application Reset	Page 8-267
OVCCON	Overlay Control	$1E4_H$	U,SV	SV, P	Application Reset	Page 8-268
–	Reserved	$1E8_H$ - $20C_H$	BE	BE	–	–
EICR0	External Input Channel Register 0	210_H	U, SV	U, SV, P	Application Reset	Page 8-205
EICR1	External Input Channel Register 1	214_H	U, SV	U, SV, P	Application Reset	Page 8-205
EICR2	External Input Channel Register 3	218_H	U, SV	U, SV, P	Application Reset	Page 8-205
EICR3	External Input Channel Register 4	$21C_H$	U, SV	U, SV, P	Application Reset	Page 8-205
EIFR	External Input Flag Register	220_H	U, SV	BE	Application Reset	Page 8-209
FMR	Flag Modification Register	224_H	U, SV	U, SV, P	Application Reset	Page 8-210
PDRR	Pattern Detection Result Register	228_H	U, SV	BE	Application Reset	Page 8-211
IGCR0	Interrupt Gating Register 0	$22C_H$	U, SV	U, SV, P	Application Reset	Page 8-212
IGCR1	Interrupt Gating Register 1	230_H	U, SV	U, SV, P	Application Reset	Page 8-212
IGCR2	Interrupt Gating Register 2	234_H	U, SV	U, SV, P	Application Reset	Page 8-212
IGCR3	Interrupt Gating Register 3	238_H	U, SV	U, SV, P	Application Reset	Page 8-212

Table 8-29 Register Overview of SCU (Offset from Main Register Base)

Short Name	Long Name	Offset Addr. 1)	Access Mode		Reset	Description See
			Read	Write		
–	Reserved	$23C_H$	BE	BE	–	–
DTSLIM	Die Temperature Sensor Limit Register	240_H	U, SV	U, SV, P	Application Reset	Page 8-224
–	Reserved	244_H - $3F4_H$	BE	BE	–	–
ACCEN1	Access Enable Register 1	$3F8_H$	U, SV	SV, SE	Application Reset	Page 8-277
ACCEN0	Access Enable Register 0	$3FC_H$	U, SV	SV, SE	Application Reset	Page 8-277

1) The absolute register address is calculated as follows:
Module Base Address + Offset Address (shown in this column)

2) Some registers marked (CPUx) are writeable only from the specified CPU. For these registers the setting of the ACCEN bits have no effect. For all other registers the ACCEN bits control access. Attempted writes by other masters or CPUs will not be successful, but no bus error will result.

9 Memory Test Unit (MTU)

All of the internal memories within the TC21x/TC22x/TC23x have a unified interface for the control of ECC (Error Correction), BIST (Built-in-Self-Test) and redundancy features. This interface can also be used to fill the memories with a predefined data value for initialization. The Memory Test Unit (MTU) controls and monitors these memory test, initialization and data integrity checking functions.

The MTU comprises two sections:

- MTU configuration registers.
- Memory Controller related registers.

The configuration registers are used to enable and disable memory test functionality. For some memories, a special sequence must be followed in order to execute memory tests without compromising data security. This sequence is ensured by simple state machines which automatically prevent test access to the memories at times when data content may be sensitive.

The Memory Controller registers are replicated for each memory bank in the device. These registers control the individual MBIST, redundancy and ECC settings for that specific memory block.

9.1 Memory Content Initialization

In security applications it is important that the MTU does not permit reading or modification of restricted memory content via the MBIST modules. In other applications (e.g. safety) it is important that modification of memory content is possible via MBIST (e.g. for ECC error injection or runtime self-test).

The TC21x/TC22x/TC23x can be configured for either of these application options via FLASH0.PROCOND.

Note that if RAMs embedded inside IP modules are not configured for initialization at boot then software initialization may be required before the IP module can be used. (E.g. GTM, ERAY modules)

9.1.1 Non-Security Applications

If FLASH0.PROCOND.RAMIN=11 then no automatic initialization of RAM content is performed on a reset and no automatic initialization of RAM content is performed when MBIST modules are enabled or disabled with MTU_MEMTEST.MEMxEN registers.

This permits the use of the MBIST controller by an application (E.g. for error injection, data modification or runtime memory testing) without unwanted corruption of memory content.

9.1.2 Security Applications

If FLASH0_PROCOND.RAMIN = 00, 01 or 10 then automatic memory content initialization is enabled. PROCOND.RAMIN configures whether the initialization is triggered by cold resets, warm resets or both. In these modes, an automatic initialization of security-sensitive memories is also triggered whenever the corresponding MTU_MEMTEST.MEMxEN is changed (i.e. The MBIST controller is enabled or disabled).

The following MBIST Controller enables are considered to be security-sensitive in this product:

- FSI0
- CPU0PTEN
- CPU0PSEN (Initializes only the cache part of the memory)
- CPU0DS2EN (Initializes only the cache part of the memory)

For security reasons, no MTU registers associated with HSM or FSI modules are accessible to the user after start-up.

9.2 Safety Notifications

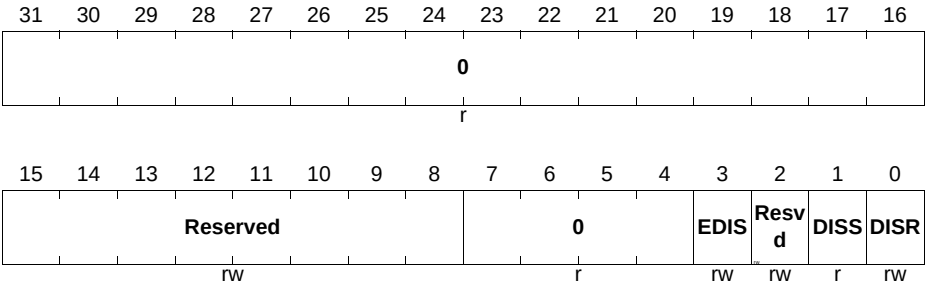
SRAM ECC errors from all system SRAMs may be reported to the Safety Management Unit (SMU). The SMU may be programmed to initiate appropriate action.

9.3 Memory Test Unit (MTU) Kernel Registers

These are the MTU control registers

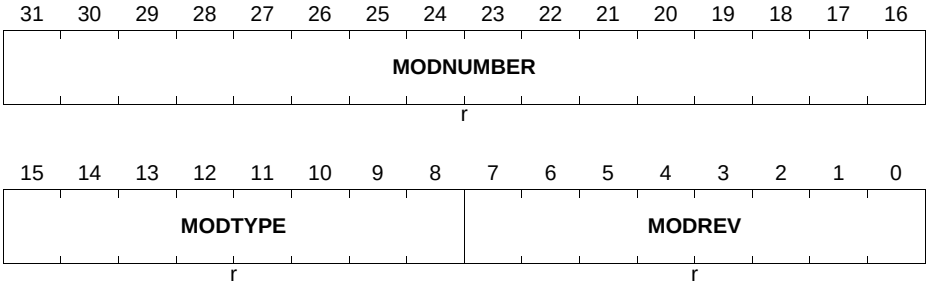
9.3.1 Descriptions

MTU_CLC
Identification Register **(F006 0000$_H$)** **Reset Value: 0000 0003$_H$**



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. 0_B Module disable is not requested 1_B Module disable is requested
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module 0_B Module is enabled 1_B Module is disabled If the RMC field is implemented and if it is 0, DISS is set automatically.
Resvd	2	rw	**Resvd** Read as 0. Must be written with 0_H
EDIS	3	rw	**Sleep Mode Enable Control** Used for module Sleep Mode control. 0_B Sleep Mode request is regarded. Module is enabled to go into Sleep Mode on a request. 1_B Sleep Mode request is disregarded: Sleep Mode cannot be entered on a request.
Reserved	[15:8]	rw	**Reserved** Read as 0. Must be written with 0_H
0	[31:16], [7:4]	r	**0** Read as 0.

MTU_ID
Identification Register (F006 0008$_H$) Reset Value: 00B2 C0XX$_H$



Field	Bits	Type	Description
MODREV	[7:0]	r	**Module Revision Number** This bit field indicates the revision number of the MTU module .
MODTYPE	[15:8]	r	**Module Type** This bit field is C0$_H$. It defines a 32-bit module
MODNUMBER	[31:16]	r	**Module Number** This bit field defines the module identification number. The idenfication number for the AURIX MTU is 00B2$_H$

The memory test register MEMTEST holds CPU configurable select bits for the various MBIST controllers.

MTU_MEMTEST0
Memory MBISTEnable Register 0 (F006 0010$_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEM31EN	**MEM30EN**	**MEM29EN**	**MEM28EN**	**MEM27EN**	**MEM26EN**	**MEM25EN**	**MEM24EN**	**MEM23EN**	**MEM22EN**	**MEM21EN**	**MEM20EN**	**MEM19EN**	**MEM18EN**	**MEM17EN**	**MEM16EN**
rwh															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEM15EN	**MEM14EN**	**MEM13EN**	**MEM12EN**	**MEM11EN**	**MEM10EN**	**MEM9EN**	**MEM8EN**	**MEM7EN**	**MEM6EN**	**MEM5EN**	**MEM4EN**	**MEM3EN**	**MEM2EN**	**MEM1EN**	**MEM0EN**
rwh															

Field	Bits	Type	Description
MEMxEN (x = 0-31)	x	rwh	**Memory x MBIST Controller Memory Test Enable** 0_B Memory x MBIST controller is disabled 1_B Memory x MBIST controller is enabled[1)] **Security Notes:** For bits which represent security-sensitive memories an automatic auto-initialization of the associated memory x is triggered on every attempt to toggle the MEMxEN [2)]. Only after this initialization has completed will the value read back from this register bit show the updated value. Register bit MEMSTAT.AIUx provides an indication that the automatic initialization of memory x is underway. See the Implementation Section for a list of memories which are security-sensitive

1)

2)

The memory test register MEMTEST holds CPU configurable select bits for the various MBIST controllers. See the Integration Section for mapping of memory controller numbers.

MTU_MEMTEST1
Memory MBISTEnable Register 1 (F006 0014$_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEM63EN	**MEM62EN**	**MEM61EN**	**MEM60EN**	**MEM59EN**	**MEM58EN**	**MEM57EN**	**MEM56EN**	**MEM55EN**	**MEM54EN**	**MEM53EN**	**MEM52EN**	**MEM51EN**	**MEM50EN**	**MEM49EN**	**MEM48EN**
rwh															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEM47EN	**MEM46EN**	**MEM45EN**	**MEM44EN**	**MEM43EN**	**MEM42EN**	**MEM41EN**	**MEM40EN**	**MEM39EN**	**MEM38EN**	**MEM37EN**	**MEM36EN**	**MEM35EN**	**MEM34EN**	**MEM33EN**	**MEM32EN**
rwh															

Field	Bits	Type	Description
MEMxEN (x = 32-63)	x-32	rwh	**Memory x MBIST Controller Memory Test Enable** 0_B Memory x MBIST controller is disabled 1_B Memory x MBIST controller is enabled[1)] **Security Notes:** For bits which represent security-sensitive memories an automatic auto-initialization of the associated memory x is triggered on every attempt to toggle the MEMxEN [2)]. Only after this initialization has completed will the value read back from this register bit show the updated value. Register bit MEMSTAT.AIUx provides an indication that the automatic initialization of memory x is underway. See the Implementation Section for a list of memories which are security-sensitive

1)

2)

The memory test register MEMTEST holds CPU configurable select bits for the various MBIST controllers. See the Integration Section for mapping of memory controller numbers.

MTU_MEMTEST2

Memory MBISTEnable Register 2 (F006 0018$_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								**MEM 87EN**	**MEM 86EN**	**MEM 85EN**	**MEM 84EN**	**MEM 83EN**	**MEM 82EN**	**MEM 81EN**	**MEM 80EN**
r								rwh							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEM 79EN	**MEM 78EN**	**MEM 77EN**	**MEM 76EN**	**MEM 75EN**	**MEM 74EN**	**MEM 73EN**	**MEM 72EN**	**MEM 71EN**	**MEM 70EN**	**MEM 69EN**	**MEM 68EN**	**MEM 67EN**	**MEM 66EN**	**MEM 65EN**	**MEM 64EN**
rwh															

Field	Bits	Type	Description
MEMxEN (x = 64-87)	x-64	rwh	**Memory x MBIST Controller Memory Test Enable** 0_B Memory x MBIST controller is disabled 1_B Memory x MBIST controller is enabled[1)] **Security Notes:** For bits which represent security-sensitive memories an automatic auto-initialization of the associated memory x is triggered on every attempt to toggle the MEMxEN [2)]. Only after this initialization has completed will the value read back from this register bit show the updated value. Register bit MEMSTAT.AIUx provides an indication that the automatic initialization of memory x is underway. See the Implementation Section for a list of memories which are security-sensitive
Res	[31:24]	r	**Reserved** Read as 0.

1)

2)

The Memory Mapping Enable register MEMMAP has configurable control bits to select memory-mapped test mode. See the Integration Section for mapping of memory controller numbers.

MTU_MEMMAP

Memory Mapping Enable Register (F006 001C$_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEM 31MAP	**MEM 30MAP**	**MEM 29MAP**	**MEM 28MAP**	**MEM 27MAP**	**MEM 26MAP**	**MEM 25MAP**	**MEM 24MAP**	**MEM 23MAP**	**MEM 22MAP**	**MEM 21MAP**	**MEM 20MAP**	**MEM 19MAP**	**MEM 18MAP**	**MEM 17MAP**	**MEM 16MAP**
rwh															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEM 15MAP	**MEM 14MAP**	**MEM 13MAP**	**MEM 12MAP**	**MEM 11MAP**	**MEM 10MAP**	**MEM 9MAP**	**MEM 8MAP**	**MEM 7MAP**	**MEM 6MAP**	**MEM 5MAP**	**MEM 4MAP**	**MEM 3MAP**	**MEM 2MAP**	**MEM 1MAP**	**MEM 0MAP**
rwh															

Field	Bits	Type	Description
MEMxMAP (x = 0-31)	x	rwh	**MEMx Mapping Enable** 0_B Memory x functional 1_B Memory x memory-mapped (e.g. for test) Note that only CPU Cache Memory Mapping bits are implemented (See Implementation Section for details of used bits in this product) **Security Notes:** Caches are considered security-sensitive memories and an automatic auto-initialization of the associated memory x is triggered on every attempt to toggle the MEMxMAP bit [1]. Only after this initialization has completed will the value read back from this register bit show the updated value. Register bit MEMSTAT.AIUx provides an indication that the automatic initialization of memory x is underway.

1)

The memory status register MEMSTAT shows whether each MBIST controller is currently executing an automatic initialization sequence.

MTU_MEMSTAT0
Memory AutoinitializeStatus Register 0 (F006 0038$_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEM 31AIU	**MEM 30AIU**	**MEM 29AIU**	**MEM 28AIU**	**MEM 27AIU**	**MEM 26AIU**	**MEM 25AIU**	**MEM 24AIU**	**MEM 23AIU**	**MEM 22AIU**	**MEM 21AIU**	**MEM 20AIU**	**MEM 19AIU**	**MEM 18AIU**	**MEM 17AIU**	**MEM 16AIU**
rh															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEM 15AIU	**MEM 14AIU**	**MEM 13AIU**	**MEM 12AIU**	**MEM 11AIU**	**MEM 10AIU**	**MEM 9AIU**	**MEM 8AIU**	**MEM 7AIU**	**MEM 6AIU**	**MEM 5AIU**	**MEM 4AIU**	**MEM 3AIU**	**MEM 2AIU**	**MEM 1AIU**	**MEM 0AIU**
rh															

Field	Bits	Type	Description
MEMxAIU (x = 0-31)	x	rh	**Memory x MBIST AutoInitialize Underway** 0_B Memory x MBIST not running autoinitialize 1_B Memory x MBIST running autoinitialize This bit indicates whether an automatic data initialization of Memory x has been triggered by a change of state of MEMTEST.MEMxEN or MEMxMAP but that the intialization sequence has not yet completed.

MTU_MEMSTAT1
Memory AutoinitializeStatus Register 1 (F006 003C$_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEM63AIU	MEM62AIU	MEM61AIU	MEM60AIU	MEM59AIU	MEM58AIU	MEM57AIU	MEM56AIU	MEM55AIU	MEM54AIU	MEM53AIU	MEM52AIU	MEM51AIU	MEM50AIU	MEM49AIU	MEM48AIU
rh															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEM47AIU	MEM46AIU	MEM45AIU	MEM44AIU	MEM43AIU	MEM42AIU	MEM41AIU	MEM40AIU	MEM39AIU	MEM38AIU	MEM37AIU	MEM36AIU	MEM35AIU	MEM34AIU	MEM33AIU	MEM32AIU
rh															

Field	Bits	Type	Description
MEMxAIU (x = 63-32)	x-32	rh	**Memory x MBIST AutoInitialize Underway** 0_B Memory x MBIST not running autoinitialize 1_B Memory x MBIST running autoinitialize This bit indicates whether an automatic data initialization of Memory x has been triggered by a change of state of MEMTEST.MEMxEN or MEMxMAP but that the intialization sequence has not yet completed.

The memory status register MEMSTAT shows whether each MBIST controller is currently executing an automatic initialization sequence.

MTU_MEMSTAT2

Memory AutoinitializeStatus Register 2 (F006 0040$_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								MEM87AIU	MEM86AIU	MEM85AIU	MEM84AIU	MEM83AIU	MEM82AIU	MEM81AIU	MEM80AIU
r								rh							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEM79AIU	MEM78AIU	MEM77AIU	MEM76AIU	MEM75AIU	MEM74AIU	MEM73AIU	MEM72AIU	MEM71AIU	MEM70AIU	MEM69AIU	MEM68AIU	MEM67AIU	MEM66AIU	MEM65AIU	MEM64AIU
rh															

Field	Bits	Type	Description
MEMxAIU (x = 87-64)	x-64	rh	**Memory x MBIST AutoInitialize Underway** 0_B Memory x MBIST not running autoinitialize 1_B Memory x MBIST running autoinitialize This bit indicates whether an automatic data initialization of Memory x has been triggered by a change of state of MEMTEST.MEMxEN or MEMxMAP but that the intialization sequence has not yet completed.
Res	[31:24]	r	**Reserved** Read as zero

MTU_RES0

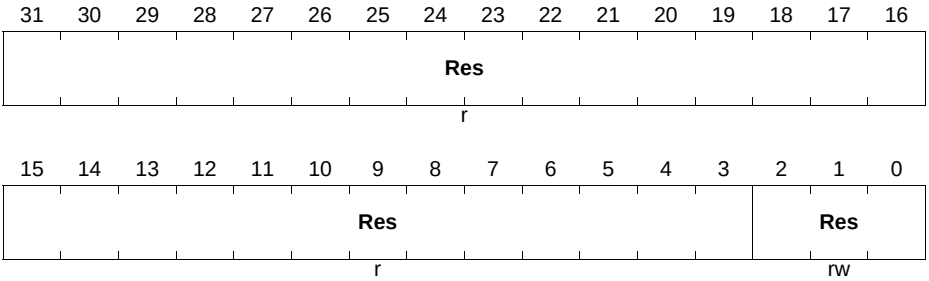
Reserved Register **(F006 0020$_H$)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res												Res	Res	Res	
r												rw	r	rw	

Field	Bits	Type	Description
Res	0,1,3	rw	**Reserved in this Product**
Res	[31:4],2	r	**Reservedin this Product**

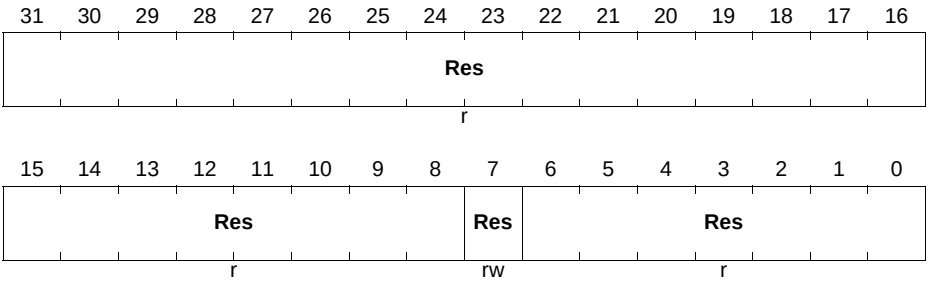
MTU_RES1
Reserved Register **(F006 0024$_H$)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
Res	[31:3]	r	**Reserved**
Res	[2:0]	rw	**Reserved in this Product**

Note:

MTU_RES2
Reserved Register **(F006 0030$_H$)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
Res	[31:8],[6:0]	r	**Reserved**
Res	7	rw	**Reserved in this Product**

MTU_RES3
Reserved Register **(F006 0034$_H$)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res															
r															

Field	Bits	Type	Description
Res	[31:0]	r	**Reserved**

The Access Enable Register 0 restricts write access to all MTU registers so that they may only be written by specified bus masters (eg CPUs). See the Bus chapter for the mapping of TAG ID to specific system masters and CPUs).

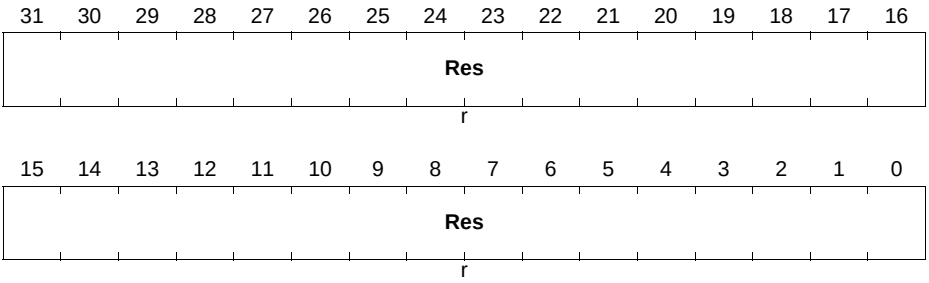
MTU_ACCEN0
Access Enable Register 0 **(F006 00FC$_H$)** **Reset Value: FFFF FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN 31	**EN 30**	**EN 29**	**EN 28**	**EN 27**	**EN 26**	**EN 25**	**EN 24**	**EN 23**	**EN 22**	**EN 21**	**EN 20**	**EN 19**	**EN 18**	**EN 17**	**EN 16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN 15	**EN 14**	**EN 13**	**EN 12**	**EN 11**	**EN 10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n = 0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the MTU kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

MTU_ACCEN1
Access Enable Register 1 **(F006 00F8$_H$)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
Res	[31:0]	r	**Reserved** Read as 0; should be written with 0.

9.3.2 MTU Register Overview

Table 9-1 Register Overview of MTU Configuration register block

Short Name	Long Name	Offset Addr.[1)]	Access Mode		Reset	Description See
			Read	Write		
MTU_CLC	Reserved	00$_H$	U,SV	SV,E,P	Application Reset	
–	Reserved	04$_H$	BE	BE	–	
MTU_ID	Identification Register	08$_H$	U, SV	BE	–	Page 9-4

Table 9-1 Register Overview of MTU Configuration register block

Short Name	Long Name	Offset Addr.[1]	Access Mode		Reset	Description See
			Read	Write		
–	Reserved	$0C_H$	BE	BE	–	
MTU_MEMTEST0	MBIST Enables 0	10_H	U, SV	SV, SE,P	Application Reset	Page 9-4
MTU_MEMTEST1	MBIST Enables 1	14_H	U, SV	SV, SE,P	Application Reset	Page 9-5
MTU_MEMTEST2	MBIST Enables 2	18_H	U, SV	SV, SE,P	Application Reset	Page 9-6
MTU_MEMMAP	Memory Mapping Control	$1C_H$	U, SV	SV, SE,P	Application Reset	Page 9-7
–	Reserved Registers	20_H-34_H	-	-	–	–
MTU_MEMSTAT0	Memory AutoInit Status Register 0	38_H	U, SV	BE	Application Reset	Page 9-8
MTU_MEMSTAT1	Memory AutoInit Status Register 1	$3C_H$	U, SV	BE	Application Reset	Page 9-9
MTU_MEMSTAT2	Memory AutoInit Status Register 2	40_H	U, SV	BE	Application Reset	Page 9-10
–	Reserved	44_H-$F4_H$	BE	BE	–	
MTU_ACCEN1	Access Enable Register 1	$F8_H$	U, SV	BE	Application Reset	Page 9-13
MTU_ACCEN0	Access Enable Register 0	FC_H	U, SV	SV, SE	Application Reset	Page 9-12

1) The absolute register address is calculated as follows:
Memory Module Register Base Address + Offset Address (shown in this column)

9.4 Memory Controllers

9.4.1 Control and Status Interfaces

9.4.1.1 Direct CPU Interface

There is a dedicated interface to the registers of each MBIST/ECC controller.

Register Mapping

Registers have an MTU system address and can be accessed through normal 16 bit SPB bus accesses. In the MTU the registers appear as if they were 16 bits wide.

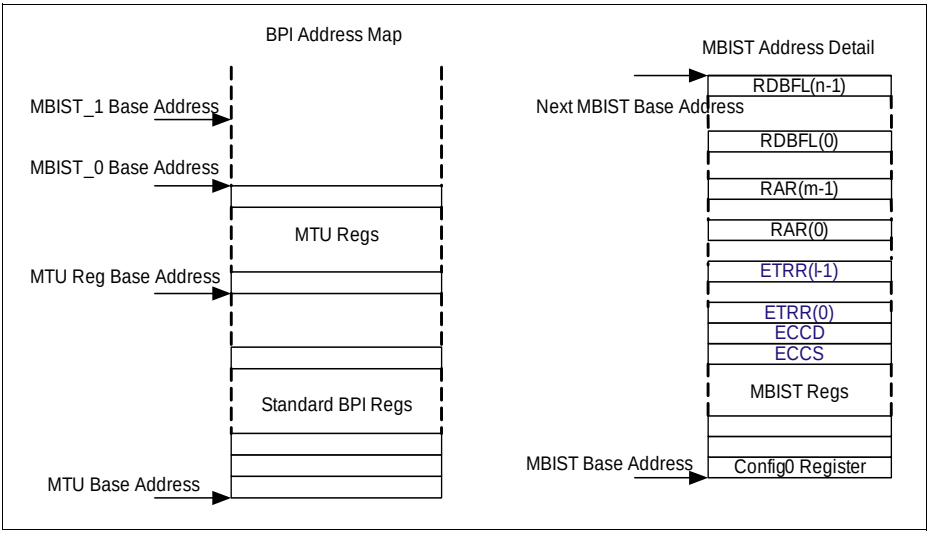


Figure 9-1 MBIST/ECC Register Mapping Scheme

9.4.2 Registers

9.4.2.1 MBIST/ECC Registers

Some register field sizes or content depend upon the physical sizes of the memories. The default settings enable fill/test of the entire physical RAM and the register descriptions show the maximum bitfield sizes.

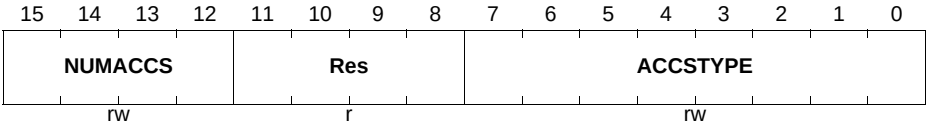
Configuration Registers

The bits in these registers can be used to control and program any march and hammer sequences. All bits concerning these test are concentrated here. All bits do not change during a test run. MCONTROL.START will start the tests defined here. MSTATUS.DONE is reset at the beginning of a test and set after completion. If no legal

operation was defined in CONFIG1.AG_MOD nothing will be done but the handshake of MCONTROL.START and MSTATUS.DONE is carried out.

The reset values of the CONFIG0/1 and MCONTROL registers will perform a {⇑(w0,r0)} operation (direction up, write 0 to all cells and check for 0) which initializes the whole memory with 0 and checks the contents if MCONTROL.START is set. This is the start sequence of many tests.

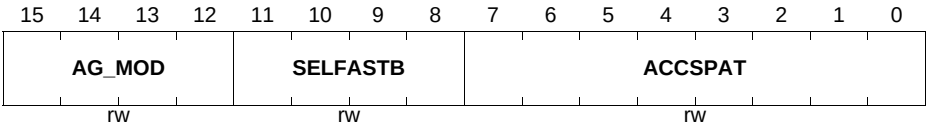
CONFIG0
Configuration Register 0 **(00_H)** **Reset Value: 2002_H**



Field	Bits	Type	Description
NUMACCS	15:12	rw	**Number of accesses per address** This field specifies the total number of accesses which are being performed to each single address in the current marching element. Range is from 0 to 15. With the sizes of ACCSPAT and ACCSTYPE given only 0 to 8 make sense. NUMACCS=0 will not access a memory.
Res	11:8	r	**Reserved** Reads return 0
ACCSTYPE	7:0	rw	**Access type** This field specifies the type of access which is being performed to each single address in the current marching element. ACCSTYPE[n] specifies n-th access of the marching element. 0_B , write access 1_B , read access

Configuration Register 1

CONFIG1
Configuration Register 1 **(02_H)** **Reset Value: 0000_H**



Field	Bits	Type	Description
AG_MOD	15:12	rw	**Address Generator Mode** These bits enable the special hardware for performing the more complex addressing schemes. In case RANGE.RAEN (range enable) is set to 0 (single access) linear address mode has to be selected and NUMACCS set to 1. 0000_B, run the test with linear address generation 0001_B, run the right half select test 0010_B, run the test with GALPAT9 algorithm 0011_B, run the left half select test 0100_B, run the test with the GALPAT5 algorithm 0101_B, run the non-destructive inversion test 1000_B, run the write mask test 1010_B, run the test with 2^i address generation **others**, Nothing is done but the handshake of START and DONE is carried out.

Field	Bits	Type	Description
SELFASTB	11:8	rw	**Select Fast Bit** This field defines during a 2^i test the address bit position that has the Hamming distance of 1, i. e. changes fastest. Bit 0 of either column or row address is swapped with the indicated bit of either column or row according to MCONTROL.RCADR. MCONTROL.RCADR=0 -> column MCONTROL.RCADR=1 -> row 0000_B, normal addressing sequence, bit 0 in its normal position. **others**, bit 0 swapped with the indicated position.
ACCSPAT	7:0	rw	**Access pattern** This field specifies directly the bit pattern which is being used for an access to each single address in the current marching element. ACCSPAT[n] specifies the n-th access of the marching element. These patterns are toggled according to BITTOG and ROWTOG.

MBIST Control Register

The bits in this register control the behavior of the MBIST, start and define tests to be performed. Any test will be performed and started when the startbit is set. The end will be signalled by MSTATUS.DONE . .

Required data (e.g. march elements, select fast bit) are taken from CONFIG0 and CONFIG1.

None of the tests are interruptable.

“Fast Row”, “Fast Column” (formerly called Fast Y and Fast X resp.) are defined as follows: “Fast Row” moves along the word-lines first and then in bit-line direction, “Fast Column” along the bit-lines first.

Note: Writes to this register are only permitted when no test is under way.

MCONTROL
MBIST Control Register **(04_H)** **Reset Value: 4008_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	**Res4**			**Res**		**FAIL DMP**	**GP_ BASE**	**BITT OG**	**ROW TOG**	**RCA DR**	**DINI T**	**DIR**	**EST F**	**RES UME**	**STA RT**
r	rw			r		rw	rw	rw	rw	rw	rw	rw	rw	rwh	rw

Field	Bits	Type	Description
Res	15	r	**Reserved** Reads return 0
Res4	14:10	r	**Reserved** Read returns 0x4 Must always be written with 0x4

Field	Bits	Type	Description
FAILDMP	9	rw	**Fail bitmap dump** This field enables a dump of the failing address and a fail bit map after a fault has been detected. The memory test is suspended afterwards and resumed by RESUME. MSTATUS.FDA shows that a fail dump is available. This functionality can be used only if bit LDRED = 1. In case a fail dump is available, RDBFL will contain the fail bit map and ETRR the failing address. ECCS/ECCD.AENE shouldn't be set at the same time as MCONTROL.FAILDUMP is set because wrong address errors are notified when a fail dump is available. 0_B Do not dump 1_B Dump each fault
GP_BASE	8	rw	**Galpat Base** This bit defines the value which is written into the base cell during the galpat mode.BITTOG and ROWTOG modify the actual value written. 0_B Write a zero into the base cell prior to reading the background pattern from the adjacent cells 1_B Write a one into the base cell prior to reading the background pattern from the adjacent cells
BITTOG	7	rw	**Bit toggling** This field specifies whether to toggle the used bit pattern (non inverted/inverted) with each physical memory column. This is required when writing a checkerboard pattern or a column stripe pattern. 0_B Do not toggle 1_B Do toggle with each column
ROWTOG	6	rw	**Row toggling** This field specifies whether to toggle the used bit pattern (non inverted/inverted) with each physical memory row. This is required when writing a checkerboard pattern or a row stripe pattern. Setting both BITTOG=1 and ROWTOG=1 would not make sense. The user must ensure that only one of these is set. 0_B Do not toggle 1_B Do toggle with each row

Field	Bits	Type	Description
RCADR	5	rw	**Fast Row / Fast Column Addressing Scheme Select** This bit selects between fast row and fast column addressing. It is compatible with the MBP 4.2 X/Y definitions. 0_B Fast row 1_B Fast column
DINIT	4	rw	**Data Initialization Enable** This bit enables a write of the RDBFL data to all locations defined by the range register. RDBFL can contain data that will produce an ECC error. Execution is started with MCONTROL.START. For this predefined action any information contained in CONFIG0/1 registers and the bits BITTOG, ROWTOG and DIR are ignored. 0_B Disabled 1_B Enabled
DIR	3	rw	**Direction Select** This field specifies the direction of a memory test operation. 0_B Address direction is highest to lowest. 1_B Address direction is lowest to highest.
ESTF	2	rw	**Enable Sticky Fail Bit** This bit enables the sticky fail bit MSTATUS.SFAIL. If set any fails will be collected in MSTATUS.SFAIL. Resetting this bit to 0 will also reset MSTATUS.SFAIL. 0_B Do not collect fail events 1_B Collect fail events
RESUME	1	rwh	**Resume failed test** This bit allows a test with fail that got suspended to be resumed after the dump of the fail bit map. A restart is possible only if MSTATUS.FDA was reset by hardware. It will be reset by hardware once the test is resumed. 0_B Do not resume 1_B Resume suspended MBIST run

Field	Bits	Type	Description
START	0	rw	**START** If this bit is written to ’1’ by software the memory test will start. If it is reset by software, and the test has finished, MSTATUS.DONE will be set to 1. If MCONTROL.FAILDMP is set, a fail will stop the current execution. RESUME will continue a suspended test. 1_B Start memory test 0_B No test started, finished or waiting for test end

Fail Dump

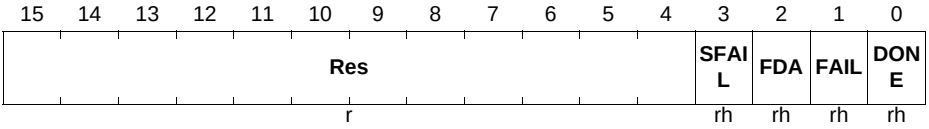
Bit FAILDMP enables a dump of the failing address and a fail bit map after a fault has been detected. The memory test is suspended afterwards and resumed by RESUME. MSTATUS.FDA shows that a fail dump is available.

This bit has to be polled by the tester hardware as there is no signal that shows a stopped test.

MBIST Status Register

The bits in the status register show the status of the currently running and last test respectively.

MSTATUS
Status Register **(06_H)** **Reset Value: 0001_H**



Field	Bits	Type	Description
Res	15:4	r	**Reserved** Reads return 0
SFAIL	3	rh	**Sticky Fail Bit** This bit is set to 1 together with FAIL provided MCONTROL.ESTF is set. In contrast to FAIL it will not be reset when a new test is started. Therefore it will collect fail information over more than one MBIST run. It will be reset when MCONTROL.ESTF is reset or the 0_B , No fail collected. 1_B , A fail occured during one of the test runs since MCONTROL.ESTF was set to 1.

Field	Bits	Type	Description
FDA	2	rh	**Fail Dump Available** This bit shows that a fail has occurred if MCONTROL.FAILDMP is set. The test is suspended and fail dump information is available. The fail bit map is in RDBFL and the associated address is in ETRR(0). As long as no fail has occurred RDBFL contains the last read information and ETRR has no valid data . This bit will be set by hardware. It will be reset when MSTATUS was read with MSTATUS.FDA = 1 and the dump information was read from ETRR and RDBFL. Only the last read from the last word of RDBFL is checked by the hardware and taken as an indication for a complete read. A suspended test will be resumed by MCONTROL.RESUME if FDA was reset. This forms some sort of handshake to insure that a suspended test can only be resumed (by a broadcasted) MCONTROL.RESUME if the last fail information was actually collected. 0_B , No fail dump data available. A suspended MBIST run can be resumed. 1_B , Fail dump data is available and waiting for read.
FAIL	1	rh	**FAIL** This bit will be reset when a test is being started. It will be set to ‘1’ by hardware under the following conditions: FAIL:=’1’ if the memory has at least one fault
DONE	0	rh	**DONE** Reset value is 1. This bit is reset at the start of a test and set when a test is completed and MCONTROL.START was reset by software. It is not set when a test is interrupted for fail dump.

Range Register

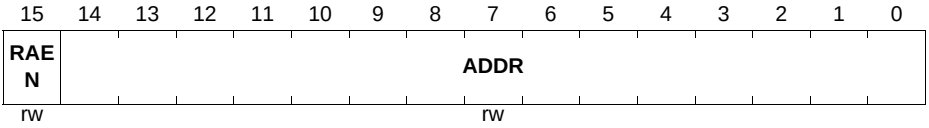
The Range Register can be used to run a test only on a dedicated part of the RAM. This can be helpful to shorten the time which is needed to find a defect cell in a FAR where there's already some information available about a failing memory cell because it's not necessary to run the test using the full address range of the RAM under test.

The range can be set in 64 word increments.

The range register can also be used to write to one specific logical address or read from it. In this case the range is disabled and the remaining part of the register is used as the logical address field.

RANGE
Range Register, single address mode (08_H)

Reset Value: $xxxx_H$



Field	Bits	Type	Description
RAEN	15	rw	**Range Enable** 0 disabled, single address mode. In this case a single word can be addressed for read or write. Config registers have to be set as follows CONFIG.NUMACCS:= "0001" (single access) CONFIG.AG_MOD := "0000" (linear) MCONTROL.DIR :=1 (up) For read just the value in this location will be delivered. No check against expected values is made; i.e. MSTATUS.FAIL will not be set.
ADDR	14:0	rw	**Address** This field specifies the logical address (before descrambling) of a single memory location. Reads and writes to this location are possible.

Setting Address Ranges

If the RAEN field is set to '1' then range mode is enabled. In this case, the ADDR field of the RANGE register is interpreted as two separate fields:

ADDR[14:7] is interpreted as the Upper Range Limit.

This field specifies the upper logical block address limit in 64 word increments.

Upper end of the address range is UPLIMIT & 111111B

ADDR[6:0] is interpreted as the Lower Range Limit

This field specifies the upper logical block address limit in 64 word increments.

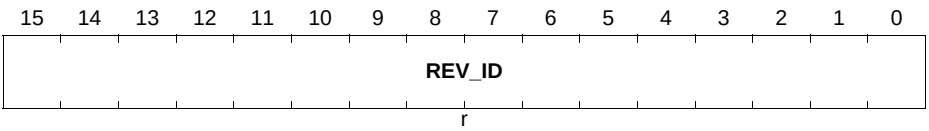
Lower end of the address range is LOLIMIT & 000000B

Note that the default reset value of the ADDR field will be set to the maximum range of the physical memory. The default behaviour is therefore that an initialization or test will operate over the whole memory.

Revision ID Register

The revision ID register contains a hard coded read only constant which describes the current status of the MBIST/ECC IP.

REVID
Revision ID Register **($0C_H$)** **Reset Value: 0510_H**



Field	Bits	Type	Description
REV_ID	15:0	r	**Revision Identifier** This field defines the currently implemented release, version and functionality of the used MBIST/ECC controller to track the MBIST/ECC version for easier handling at the tester.

ECC Safety Register

This register controls the various error detection and notification modes.

Writes to this register are only permitted when safe_endinit= 0 and no test is under way.

The hardware features that implement fault tolerance mechanisms for the SRAMs (e.g. single bit correction) shall be enabled per default after any reset. This is ensured by the reset value of the ECC safety register where ECCS.ECE = 1 after a reset.

ECCS
ECC Safety Register **($0E_H$)** **Reset Value: $00XF_H$**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				TC_WAY_SEL		ECCMAP		SFLE		BFLE	TRE	ECE	AENE	UENE	CENE
r				rw		rw		rw		rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
Res	15:12	r	**Reserved** Reads return 0
TC_WAY_SEL	11:10	rw	**TriCore Cache Way Select** TriCore Cache only. These two bits select a cache way to run the non-destructive inversion test on. The bits represent the way number.
ECCMAP	9:8	rw	**ECC Bit Mapping Mode** ECCMAP sets three different test modes to allow access to data or ECC bits separately and independently. 00_B Normal operation 01_B Test mode. Only data bits mapped. All ECC functionality disabled. 10_B Test mode. ECC check bits mapped to lower data bit positions. Other bits read as zero. All ECC functionality disabled. Data bits are not affected by write operations. 11_B not allowed.

Field	Bits	Type	Description
SFLE	7:6	rw	**Signature Bit Flip Enables** If any SFLE bit is set no error tracking will take place. 00_B Normal operation 01_B Test mode only. Flips bitline address check signature bits. 10_B Test mode only. Flips wordline address check signature bits. 11_B Test mode only. Combination makes no sense.
BFLE	5	rw	**Bit Flip Enable** 0_B Normal operation 1_B Test mode only. Flips data and check bits according to RDBFL.
TRE	4	rw	**Tracking Enable** All errors will be tracked, if the associated notification enable bit is set. For TriCore Memory MC Modules [1] this bit is set by default 0_B Do not track address of detected error 1_B Track address of detected error
ECE	3	rw	**Error Correction Enable** This bit shadows ECCD.ECE. Either bit can be written and both bits will read the last written value. 0_B Do not correct correctable errors 1_B Correct correctable errors
AENE	2	rw	**Address Error Notification Enable** This bit shadows ECCD.AENE. Either bit can be written and both bits will read the last written value. 0_B Do not report address errors 1_B Report detected address errors
UENE	1	rw	**Uncorrectable Error Notification Enable** This bit shadows ECCD.UENE. Either bit can be written and both bits will read the last written value. 0_B Do not report uncorrectable data errors 1_B Report detected uncorrectable errors to SMU

Field	Bits	Type	Description
CENE	0	rw	**Correctable Error Notification Enable** This bit shadows ECCD.CENE. Either bit can be written and both bits will read the last written value. 0_B Do not report correctable data errors 1_B Report detected correctable errors to SMU

1) A special reset value of ‘1’ is implemented for TRE bit of MCs of all TriCore Memories. TriCore Memories means all DTAG, PTAG, PSPR, DSPR and DSPR2 Memory Controllers (if present in product)

ECC Detection Register

The ECC detection register contains information on the errors detected and the tracking register clear.

Bits EOV, AERR, UERR and CERR are signals as well. While the bits are cleared by software write, the signals have a handshake associated to cater for different clock domains.

ECCD
Memory ECC Detection Register (10_H) Reset Value: 7800_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOV	**ECE**	**AENE**	**UENE**	**CENE**	**Res**	**VAL**					**TRC**	**AERR**	**UERR**	**CERR**	**SERR**
rh	rw	rw	rw	rw	rwh	rh					w	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
EOV	15	rh	**Error Overflow** 0_B all errors detected since last clear were tracked. 1_B more errors were detected since last clear than error tracking registers are available. Also, this bit is set if more than one memory block was in error at the same time. See ETRR.MBI for details. Enabled by ECCS.TRE and reset by ECCD.TRC
ECE	14	rw	**Error Correction Enable** Note that this bit is only writeable when Safety Endinit = 0. Other writes will not change this bit. 0_B Do not correct correctable errors 1_B Correct correctable errors

Field	Bits	Type	Description
AENE	13	rw	**Address Error Notification Enable** This bit enables ECCD.AERR and the hardware flag to SMU. Note that this bit is only writeable when Safety Endinit = 0. Other writes will not change this bit. AENE shouldn't be set at the same time as MCONTROL.FAILDUMP is set because wrong address errors are notified when a fail dump is available. 0_B Do not report address errors 1_B Report detected address errors
UENE	12	rw	**Uncorrectable Error Notification Enable** This bit enables ECCD.UERR and the hardware flag to SMU. Note that this bit is only writeable when Safety Endinit = 0. Other writes will not change this bit. 0_B Do not report uncorrectable data errors 1_B Report detected uncorrectable errors to SMU
CENE	11	rw	**Correctable Error Notification Enable** This bit enables ECCD.CERR and the hardware flag to SMU. Note that this bit is only writeable when Safety Endinit = 0. Other writes will not change this bit. 0_B Do not report correctable data errors 1_B Report detected correctable errors to SMU
VAL	9:5	rh	**Valid Bits** Every tracking register has a valid bit associated. Reset by ECCD.TRC. Up to five error tracking registers are available and five valid bits. Unused valid bits always read 0.
TRC	4	w	**Tracking Clear** Writing this bit with '1' clears the EOV, VAL bits plus the tracking registers. This bit will always read 0. 0_B No effect 1_B Clear bits

Field	Bits	Type	Description
AERR	3	rwh	**Address Error Detected** Write of '0' clears the sticky status. Write of '1' has no effect. A write does not clear the hardware flag.[1)] Read as: 0_B No address error detected 1_B Address error detected (AERR)
UERR	2	rwh	**Uncorrectable Error Detected** Write of '0' clears the sticky status. Write of '1' has no effect. A write does not clear the associated hardware flag. Read as: 0_B No uncorrectable error detected 1_B Uncorrectable error detected (UERR)
CERR	1	rwh	**Correctable Error Detected** Write of '0' clears the sticky status. Write of '1' has no effect. A write does not clear the associated hardware flag. Read as: 0_B No correctable error detected 1_B Correctable error detected (CERR)
SERR	0	rwh	**Error Detected** Write of '0' clears the sticky status. Write of '1' has no effect. Read as: 0_B No error detected 1_B Error detected: any error detected :CERR, UERR or AERR

1) Note: A write to AERR, UERR and CERR does not clear the hardware flags.

Error Tracking Register(s)

This/these register(s) contain(s) the address of error(s) detected. Up to five registers are implemented per MBIST according to VHDL generic nb_etrr_buf_g (theoretically there is address space for 8 registers). Default value of the generic is 1. ETRR(0) contains the last error detected. If more than one register is implemented, successive errors will push previous errors up. Correctable, uncorrectable and address errors are stored here in the same manner. Only new errors will be taken into consideration. I. e. errors at stored addresses will be ignored, no matter whether the cause is the same or different from the stored address.

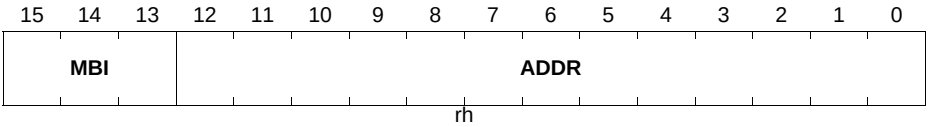
The address tracked is physical i.e. the address is directly equivalent to the SRAM address input signals a_i.

The associated valid bit(s) is/are contained in the register ECCD.VAL. ECCD.TRC clears the associated valid bits and all contents of ETRR will also be cleared.

It/they contain(s) the faulty address(es) of both the correctable, uncorrectable case and address errors.

Once all registers are used up ECCD.EOV is set .

ETRRx (x=0-4)
Error Tracking Register x **(12_H+x*02_H)** **Reset Value: 0000_H**



Field	Bits	Type	Description
MBI	15:13	rh	**Memory Block Index of Error(i)** If f more than one memory is implemented in parallel, these three bits contain the index of the memory block in error to identify the memory in error and the tracked address belongs to this memory. Otherwise these bits always are set to 0. Only one error can be tracked at one time. If more than one error occurs at the same time ECCD.EOV will be set, no matter how many ETRR registers are still available and the lowest memory index is stored.
ADDR	12:0	rh	**Address of Error(i)** Address of the error detected since last clear operation.

Read Data and Bit Flip Register

This register is used for several purposes whenever a register with the size of the memory width is needed.

Normally this register contains the data which are directly read back from the RAM (without any data scrambling) during the last read access.

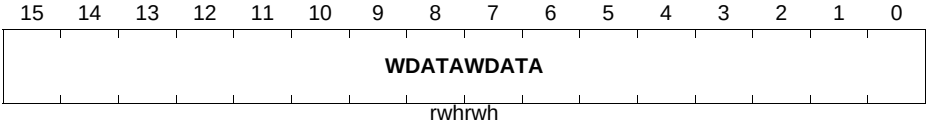
During logic test (LBIST) it contains the bit flip information. If ECCS.BFLE is set, this information is used to flip bits written to the SRAM.

After a failed test it contains the failed bit map if MCONTROL.FAILDMP is set. The address part is contained in the error tracking register.

When MCONTROL.DINIT is set, this information is written to all locations within range.

Writes to this register are not permitted while a test is underway.

RDBFLy (y=0-39)
Read Data and Bit Flip Register y ($A0_H$+ y*02_H) **Reset Value: 0000_H**



Field	Bits	Type	Description
WDATA	15:0	rwh	**Word Data** This field contains the data of the last memory read operation. The width of WDATA does always match the memory data width. m = nb_mems_g * (n + k)

9.4.3 Safety Features

Several Safety features are implemented in conjunction with ECC and Safety SRAMs.

There are three signals to provide an alarm to the SMU:

correctable error detected

uncorrectabe error detected

address error detected. Address errors are detected by the Safety SRAM feature. This discovers all wordline failures including broken wordlines and decoder malfunctions.

The codes used for ECC are of the SECDED (single error correction double error detection) type..

All codes are able to detect all zeroes and all ones, despite the fact that all physical faults resulting in this condition are covered by the Safety SRAM feature.

Whenever a new error is detected, not only a signal is set as an alarm, also an associated bit is set in ECCD. Three bits available are ECCD.CERR, ECCD.UERR and ECCD.AERR. They are cleared by writing a 0 to their location.

Bit ECE switches error correction on and off.

All errors are being tracked in error tracking registers (1 to 4). Correctable and uncorrectable errors are treated the same way except for the alarms and notificaton bits.

If an error is tracked then the associated register contains a valid value and a valid bit is set ECCD.VAL

The valid bits can be cleared by writing to ECCD.TRC

A valid bit is associated with each error entry. Bit ECCD.EOV is set when the error tracking register has recorded the maximum number of errors already and another error gets detected before the error tracking register is read by software and the event cleared by ECCD.TRC. All errors are treated the same way.

.

Uncorrectable errors are treated the same way as correctable errors and stored in the same registers.

Also, valid bits are used the same way.

ECCS.ECCMAP allows to select separately either the data bits of the stored information or the ECC check bits or data bits with ECC bits operational. In normal mode (ECCS.ECCMAP = 00) only access to data with correctly enabled ECC is possible and the ECC bits as such are neither visible nor accessible.

This mode (ECCS.ECCMAP ≠ “00”) cannot be left without prior changing safe_endinit.

Bit will switch on special test modes to inject errors into the ECC encoder to ECC decoder path. This way the redundancy of this path can be broken and the logic tested. Also, the ECC can be tested by firmware by writing illegal codes into the ECC data fields. Then any bit combination can be written to the ECC bits.

There is a Safety SRAM feature implemented to detect errors that involve more errors that can be detected by ECC. Also, there are signature bit flip enables provided to test the logic of address error detection.

The Safety SRAM feature will also detect the failure mode where a valid SRAM access request present at the SRAM input is not processed by the SRAM, as long as the attempted access does not have the same address as the last functioning valid access to the SRAM

The Safety SRAM feature detects addressing faults for both read and write accesses. Atomic read-modify-write sequences are also considered.

When such an error occurs or when a bit flip enable is set, an address error is flagged, provided ECCS.AENE is set.

Normally SRAM, MBIST and ECC are delivered as block. If the ECC decoder and/or decoder are embedded in a pipeline, then the blocks can be separated.

9.4.4 Operation Modes

9.4.4.1 Starting a Memory Test Sequence (example)

It's not possible any longer to start simple standard tests such as Checkerboard, Row Stripe, March C+ and March U without writing special values to the CONFIG registers. These tests now also require to write dedicated values into the CONFIG registers as described in the following example:

1. Enter memory test mode
2. Initialize registers
   Write the desired march sequence into the CONFIG registers
   CONFIG1.ACCSPAT is the pattern to be executed, CONFIG0.ACCSTYPE the read/write sequence and CONFIG0.NUMACCS the number of bits in CONFIG1.ACCSPAT and CONFIG0.ACCSTYPE to used
   AG_MOD contains the address mode to be used
   RANGE := <memory range to be tested>
3. Start memory test by writing a 1 to register bit MCONTROL.START
4. Reset MCONTROL.START
5. Wait for the end of the test
   - poll MSTATUS.DONE
   - or poll primary output DONE
6. Check the result of the test
   - verify MSTATUS.FAIL
   - or verify primary output FAIL

Complex algorithm may require several march starts to get a full test.

9.4.4.2 Getting Detailed Memory Test Results

The FAIL and DONE bits provide a general pass/fail information.

If MCONTROL.FAILDMP = '1', the test stops after a failed test and the fail information is immediately available for dump. MSTATUS.FDA (fail dump available) is set and signal faildumpreq asserted in this case. RDBFL contains the fail bit map and ETRR the failed address. Reading MSTATUS with MSTATUS.FDA = '1' and RDBFL(n-1) will reset MSTATUS.FDA and faildumpreq. A consequent setting of MCONTROL.RESUME will restart the interrupted test sequence.

The RANGE register can be used to run consecutive tests on constantly shifted memory ranges so that in the end the complete memory has been analyzed.

9.4.4.3 Dumping Fail Bitmap

Any dump information has to be polled from registers RDBFL and ETRR(0).

9.4.4.4 Filling a Memory with Defined Contents

The MBIST/ECC can be used to fill a memory range or a complete memory with a defined pattern very fast, i.e. one write access per cycle with the full memory data width. There are three methods implemented, one using bit MCONTROL.DINIT, one using input mtu_auto_data_init and another one using input mtu_partial_erase_i.

MCONTROL.DINIT will write a range with any pattern in RDBFL no matter whether this is a valid ECC code or not. This is useful if memories have additional ECC or parity bits and must be initialized before they can be used. The necessary steps are:

1. Enter memory test mode
2. Initialize registers
   RDBFL := <desired pattern>
   RANGE := <memory range to be filled>
3. Start fill operation by setting MCONTROL.DINIT and MCONTROL.START of the MCONTROL register
4. Wait for MSTATUS.DONE to be reset and clear MCONTROL.START.
5. Wait for the end of fill operation by polling MSTATUS.DONE bit
6. Leave memory test mode.

Also, two hardware signals named mtu_auto_data_init_i and mtu_partial_erase_i are available to fill the whole memory with the data on d_i or erase parts of the memory. ECC codes are automatically correct. Range is ignored if mtu_auto_data_init_i is applied, in case mtu_partial_erase_i is used nb_partial_erase_size_g blocks of 1kB starting from the top of the memory are erased. Memory testmode does not need to be switched on but a clock must be available.

Both signals will have priority over MCONTROL.START and are not interruptable.

9.4.4.5 Reading a Single Memory Location

The MBIST/ECC can also be used to read the contents of a single word. The RDBFL register holds the contents of a complete memory word and thus it is possible to read all memory bits, including ECC or parity bits. The necessary steps are:

1. Enter memory test mode
2. Initialize registers
   RANGE := RAEN = 0 (range disabled = single address) & address to be read
   CONFIG0 := 1001_H (NUMACCS = 1_H, ACCSTYPE = 01_H (read))
   CONFIG1 := 0000_H (linear mode, non inverted pattern)
3. MCONTROL := 4009_H (FAILDMP = 0, direction up, start): Start read operation
4. MCONTROL := 4008_H (clear START)
5. Wait for MSTATUS.DONE to be reset
6. Read RDBFL register
7. Leave memory test mode

9.4.4.6 Writing to a Single Memory Location

The MBIST/ECC can also be used to write the contents of RDBFL register to a single memory location. RDBFL holds the contents of a complete memory word and thus it is possible to write to all memory bits, including ECC or parity bits. The necessary steps are:

1. Enter memory test mode
2. Initialize registers
   RDBFL := write data
   RANGE := RAEN= 0 (range disabled = single address) & address to be written to
   CONFIG0 := 1000_H (NUMACCS = 1_H, ACCSTYPE = 00_H (write))
   CONFIG1 := 0000_H (linear mode, non inverted pattern)
3. MCONTROL := 4009_H (FAILDMP = 0, direction up, start): Start write operation
4. MCONTROL := 4008_H (clear START)
5. Wait for MSTATUS.DONE to be reset
6. Leave memory test mode

9.4.5 Memory Controller Register Addresses

There is one set of Memory Controller Registers for each logical memory module in the TC21x/TC22x/TC23x.

The sets of registers are located at offsets from the following base addresses.

Table 9-2 Registers Address Space - Memory Controller Registers

Module	Base Address	End Address	Note
MC0	$F006\ 1000_H$	$F006\ 10FF_H$	-
MC1	$F006\ 1100_H$	$F006\ 11FF_H$	-
MC2	$F006\ 1200_H$	$F006\ 12FF_H$	-
MC3	$F006\ 1300_H$	$F006\ 13FF_H$	-
MC4	$F006\ 1400_H$	$F006\ 14FF_H$	-
MC5	$F006\ 1500_H$	$F006\ 15FF_H$	-
MC6	$F006\ 1600_H$	$F006\ 16FF_H$	-
MC7	$F006\ 1700_H$	$F006\ 17FF_H$	-
MC8	$F006\ 1800_H$	$F006\ 18FF_H$	-
MC9	$F006\ 1900_H$	$F006\ 19FF_H$	-
MC10	$F006\ 1A00_H$	$F006\ 1AFF_H$	-
MC11	$F006\ 1B00_H$	$F006\ 1BFF_H$	-
MC12	$F006\ 1C00_H$	$F006\ 1CFF_H$	-
MC13	$F006\ 1D00_H$	$F006\ 1DFF_H$	-
MC14	$F006\ 1E00_H$	$F006\ 1EFF_H$	-
MC15	$F006\ 1F00_H$	$F006\ 1FFF_H$	-
MC16	$F006\ 2000_H$	$F006\ 20FF_H$	-
MC17	$F006\ 2100_H$	$F006\ 21FF_H$	-
MC18	$F006\ 2200_H$	$F006\ 22FF_H$	-
MC19	$F006\ 2300_H$	$F006\ 23FF_H$	-
MC20	$F006\ 2400_H$	$F006\ 24FF_H$	-
MC21	$F006\ 2500_H$	$F006\ 25FF_H$	-
MC22	$F006\ 2600_H$	$F006\ 26FF_H$	-
MC23	$F006\ 2700_H$	$F006\ 27FF_H$	-
MC24	$F006\ 2800_H$	$F006\ 28FF_H$	-
MC25	$F006\ 2900_H$	$F006\ 29FF_H$	-

Table 9-2 Registers Address Space - Memory Controller Registers (cont'd)

Module	Base Address	End Address	Note
MC26	F006 2A00$_H$	F006 2AFF$_H$	-
MC27	F006 2B00$_H$	F006 2BFF$_H$	-
MC28	F006 2C00$_H$	F006 2CFF$_H$	-
MC29	F006 2D00$_H$	F006 2DFF$_H$	-
MC30	F006 2E00$_H$	F006 2EFF$_H$	-
MC31	F006 2F00$_H$	F006 2FFF$_H$	-
MC32	F006 3000$_H$	F006 30FF$_H$	-
MC33	F006 3100$_H$	F006 31FF$_H$	-
MC34	F006 3200$_H$	F006 32FF$_H$	-
MC35	F006 3300$_H$	F006 33FF$_H$	-
MC36	F006 3400$_H$	F006 34FF$_H$	-
MC37	F006 3500$_H$	F006 35FF$_H$	-
MC38	F006 3600$_H$	F006 36FF$_H$	-
MC39	F006 3700$_H$	F006 37FF$_H$	-
MC40	F006 3800$_H$	F006 38FF$_H$	-
MC41	F006 3900$_H$	F006 39FF$_H$	-
MC42	F006 3A00$_H$	F006 3AFF$_H$	-
MC43	F006 3B00$_H$	F006 3BFF$_H$	-
MC44	F006 3C00$_H$	F006 3CFF$_H$	-
MC45	F006 3D00$_H$	F006 3DFF$_H$	-
MC46	F006 3E00$_H$	F006 3EFF$_H$	-
MC47	F006 3F00$_H$	F006 3FFF$_H$	-
MC48	F006 4000$_H$	F006 40FF$_H$	-
MC49	F006 4100$_H$	F006 41FF$_H$	-
MC50	F006 4200$_H$	F006 42FF$_H$	-
MC51	F006 4300$_H$	F006 43FF$_H$	-
MC52	F006 4400$_H$	F006 44FF$_H$	-
MC53	F006 4500$_H$	F006 45FF$_H$	-
MC54	F006 4600$_H$	F006 46FF$_H$	-
MC55	F006 4700$_H$	F006 47FF$_H$	-

Table 9-2 Registers Address Space - Memory Controller Registers (cont'd)

Module	Base Address	End Address	Note
MC56	$F006\ 4800_H$	$F006\ 48FF_H$	-
MC57	$F006\ 4900_H$	$F006\ 49FF_H$	-
MC58	$F006\ 4A00_H$	$F006\ 4AFF_H$	-
MC59	$F006\ 4B00_H$	$F006\ 4BFF_H$	-
MC60	$F006\ 4C00_H$	$F006\ 4CFF_H$	-
MC61	$F006\ 4D00_H$	$F006\ 4DFF_H$	-
MC62	$F006\ 4E00_H$	$F006\ 4EFF_H$	-
MC63	$F006\ 4F00_H$	$F006\ 4FFF_H$	-
MC64	$F006\ 5000_H$	$F006\ 50FF_H$	-
MC65	$F006\ 5100_H$	$F006\ 51FF_H$	-
MC66	$F006\ 5200_H$	$F006\ 52FF_H$	-
MC67	$F006\ 5300_H$	$F006\ 53FF_H$	-
MC68	$F006\ 5400_H$	$F006\ 54FF_H$	-
MC69	$F006\ 5500_H$	$F006\ 55FF_H$	-
MC70	$F006\ 5600_H$	$F006\ 56FF_H$	-
MC71	$F006\ 5700_H$	$F006\ 57FF_H$	-
MC72	$F006\ 5800_H$	$F006\ 58FF_H$	-
MC73	$F006\ 5900_H$	$F006\ 59FF_H$	-
MC74	$F006\ 5A00_H$	$F006\ 5AFF_H$	-
MC75	$F006\ 5B00_H$	$F006\ 5BFF_H$	-
MC76	$F006\ 5C00_H$	$F006\ 5CFF_H$	-
MC77	$F006\ 5D00_H$	$F006\ 5DFF_H$	-
MC78	$F006\ 5E00_H$	$F006\ 5EFF_H$	-
MC79	$F006\ 5F00_H$	$F006\ 5FFF_H$	-
MC80	$F006\ 6000_H$	$F006\ 60FF_H$	-
MC81	$F006\ 6100_H$	$F006\ 61FF_H$	-
MC82	$F006\ 6200_H$	$F006\ 62FF_H$	-
MC83	$F006\ 6300_H$	$F006\ 63FF_H$	-
MC84	$F006\ 6400_H$	$F006\ 64FF_H$	-
MC85	$F006\ 6500_H$	$F006\ 65FF_H$	-

Table 9-2 Registers Address Space - Memory Controller Registers (cont'd)

Module	Base Address	End Address	Note
MC86	$F006\ 6600_H$	$F006\ 66FF_H$	-
MC87	$F006\ 6700_H$	$F006\ 67FF_H$	-

9.4.6 Memory Controller Register Overview

This section describes registers in each memory controller module. Register names described in this section will be referenced in other parts of the TC21x/TC22x/TC23x User's Manual by the prefix "MCx_" where x is the index or name of the specific memory controller. The registers are replicated for each memory controller.

MTU Memory Control Kernel Register Overview

Table 9-3 Register Overview of each MTU Memory Control register block

Short Name	Long Name	Offset Addr.[1)]	Access Mode		Reset	Description See
			Read	Write		
CONFIG0	Memory Config Register 0	00_H	U, SV, 16	U, SV, 16		Page 9-18
CONFIG1	Memory Config Register 1	02_H	U, SV, 16	U, SV, 16	-	Page 9-19
MCONTROL	Memory Control Register	04_H	U, SV, 16	SV, SE, 16	-	Page 9-21
MSTATUS	Memory Status Register	06_H	U, SV, 16	BE	-	Page 9-25
RANGE	Memory Range Register	08_H	U, SV, 16	U, SV, 16	-	Page 9-27
–	Reserved	$0A_H$	BE	BE	–	–
REVID	Revision Register	$0C_H$	U, SV, 16	BE	-	Page 9-28
ECCS	Range Register	$0E_H$	U, SV, 16	SV, SE, 16	-	Page 9-29
ECCD	ECC Detection Register	10_H	U, SV, 16	SV, 16	-	Page 9-32
ETRRI	Error Tracking Register I (I= 0-4)	$12_H + (I * 02_H)$	U, SV, 16	BE	-	Page 9-35

Table 9-3 Register Overview of each MTU Memory Control register block

Short Name	Long Name	Offset Addr.[1]	Access Mode		Reset	Description See
			Read	Write		
–	Reserved	(14_H+(l * 02_H))-$1E_H$	BE	BE	-	-
–	Reserved	20_H-$9F_H$	BE	BE	–	–
RDBFLy	Memory Read Data Register y (y = 0-39)	(y * 02_H) + $A0_H$	U, SV, 16	U, SV, 16	-	Page 9-36
–	Reserved	$F0_H$-FF_H	BE	BE	–	–

1) The absolute register address is calculated as follows:
Memory Module Register Base Address + Offset Address (shown in this column)

9.5 ECC Implementation

9.5.1 ECC

9.5.1.1 ECC Codes

Most ECC codes except for the cache tag memories are of the SECDED (single error correction and double error detection) type. These are specially designed Hsiao codes optimized for triple error detection. They will detect all correctable errors and allow correction, detect all double errors and detect a relatively high percentage of triple errors. The cache tag memories will get DED codes (double error detection).

All codes will detect “all zeros” and “all ones”. However, with the safety SRAM concept of address error detection this feature is not required anymore as all physical faults that might produce an “all ones” or “all zeros” error are covered there.

SECDED Codes

14 bits total length, 8 bits data, 6 check bits

Matrix: [1, 2, 4, 8, 16, 32, 7, 11, 19, 35, 13, 22, 44, 52]

Check part of the matrix as bits:

00010011

00100101

01001010

10001111

11110100

11111000

No double columns, can be used as SEC code.

Type: Hsia-Code (all columns have odd weight)

Table 9-4 Code Characteristics 8/6

Number of codes with weight 3	0
Number of codes with weight 4	34
Can be used as SECDED code	
Undetectable double errors (correction on)	0
Undetectable triple errors (correction on)	136
Percentage of undetectable triple errors (correction on)	37.362637
Row weights	[6, 6, 6, 4, 4, 4]
Number of ones in the matrix	30

22 bits total length, 16 bits data, 6 check bits

Matrix: [1, 2, 4, 8, 16, 32, 7, 11, 19, 35, 13, 21, 37, 25, 41, 14, 22, 38, 26, 50, 44, 52]

Check part of the matrix as bits:

0001001010010111

0010010100101101

0100100111001010

1000111001110011

1111000001111100

1111111110000000

No double columns, can be used as SEC code.

Type: Hsia-Code (all columns have odd weight)

Table 9-5 Code Characteristics 16/6

Number of codes with weight 3	0
Number of codes with weight 4	252
Undetectable double errors (correction on)	0
Undetectable triple errors (correction on)	1008
Percentage of undetectable triple errors (correction on)	64.454544
Row weights	[10, 10, 10, 8, 8, 8]
Number of ones in the matrix	54

28 bits total length, 21 bits data, 7 check bits

Matrix: [1, 2, 4, 8, 16, 32, 64, 76, 13, 35, 52, 69, 88, 28, 49, 38, 19, 73, 84, 11, 56, 50, 22, 97, 14, 67, 98, 104]

Check part of the matrix as bits:

10001100001100001011 1

00110001100001101001 1

00010111010101110000 0

11000110001011000100 1

11011010100100010100 0

00100000110010110111 0

01101001011010001010 0

No double columns, can be used as SEC code.

Type: Hsia-Code (all columns have odd weight)

Table 9-6 Code Characteristics 21/7

Number of codes with weight 3	0
Number of codes with weight 4	1316
Undetectable double errors (correction on)	0
Undetectable triple errors (correction on)	1316

Table 9-6 Code Characteristics 21/7

Percentage of undetectable triple errors (correction on)	40.17094
Row weights	[10, 10, 10, 10, 10, 10, 10]
Number of ones in the matrix	70

31 bits total lenght, 24 bits data, 7 check bits

Maxtrix: [1, 2, 4, 8, 16, 32, 64, 15, 19, 97, 111, 41, 35, 98, 25, 104, 78, 37, 100, 112, 21, 49, 50, 84, 67, 69, 52, 73, 56, 82, 88]

Check part of the matrix as bits:

0011001011011000011101011

0011111010111011000101000

0100000100000111110010111

1001100111000000000001101

1001000001110100101100000

1101011001000001010000010

1111110100100110011010000

No double columns, can be used as SEC code.

Type: No Hsia-Code (at leat one row has even weight)

Table 9-7 Code Characteristics 24/7

Number of codes with weight 3	0
Number of codes with weight 4	442
Can be used as SECDED code	
Undetectable double errors (correction on)	0
Undetectable triple errors (correction on)	1768
Percentage of undetectable triple errors (correction on)	39.332592
Row weights	[14, 10, 10, 10, 12, 14, 14]
Number of ones in the matrix	84

36 bits total length, 29 bits data, 7 check bits

Matrix: [1, 2, 4, 8, 16, 32, 64, 7, 11, 19, 35, 67, 13, 21, 37, 69, 25, 41, 73, 49, 14, 22, 38, 70, 26, 42, 82, 98, 28, 76, 52, 84, 56, 88, 104, 112]

Check part of the matrix as bits:

00001000100100001001101010111

00010001001010010010100101011

00100010010010100101010111101

01000100011101000110011001110

10000111100001111000011110000

11111000000001111111100000000

11111111111110000000000000000

No double columns, can be used as SEC code.

Type: Hsia-Code (all columns have odd weight)

Table 9-8 Code Characteristics 29/7

Number of codes with weight 3	0
Number of codes with weight 4	979
Undetectable double errors (correction on)	0
Undetectable triple errors (correction on)	3916
Percentage of undetectable triple errors (correction on)	54.84594
Row weights	[14, 14, 14, 14, 14, 12, 12]
Number of ones in the matrix	94

39 bits total length, 32 bits data, 7 check bits

NON-INVERTIBLE code!

Matrix: [1, 2, 4, 8, 16, 32, 64, 11, 35, 67, 13, 21, 37, 69, 25, 41, 73, 49, 81, 97, 14, 22, 38, 70, 26, 42, 74, 50, 82, 98, 28, 44, 76, 52, 84, 100, 56, 88, 112]

Check part of the matrix as bits:

00100010010110001001011001011011

01000100101010010010101010101101

00001001001100100100110100110111

10010001110001000111000111000110

00011110000001111000000111111000
11100000000001111111111000000000
11111111111110000000000000000000

No double columns, can be used as SEC code.

Type: Hsia-Code (all columns have odd weight)

Table 9-9 Code Characteristics 32/7

Number of codes with weight 3	0
Number of codes with weight 4	1363
Can be used as SECDED code	
Undetectable double errors (correction on)	0
Undetectable triple errors (correction on)	5452
Percentage of undetectable triple errors (correction on)	59.656418
Row weights	[14, 14, 15, 15, 15, 15, 15]
Number of ones in the matrix	103

40 bits total length, 32 bits data, 8 check bits

Matrix: [1, 2, 4, 8, 16, 32, 64, 128, 161, 98, 35, 194, 14, 11, 38, 41, 69, 134, 25, 52, 176, 138, 196, 88, 19, 145, 67, 74, 97, 26, 164, 73, 162, 152, 148, 13, 44, 21, 84, 76]

Check part of the matrix as bits:

1001000001001110010000101110000 0
0101000010000011001110010000001 1
1110001100011000000010101000100 0
0000000000111001110001000110011 0
0000110100100101000101010101100 1
0000101011010010000000100011111 1
0111111001000100101101001000000 0
1010010110100000111010010001010 0

No double columns, can be used as SEC code.

Type: Hsia-Code (all columns have odd weight)

Table 9-10 Code Characteristics 32/8

Number of codes with weight 3	0
Number of codes with weight 4	766
Undetectable double errors (correction on)	0
Undetectable triple errors (correction on)	3064
Percentage of undetectable triple errors (correction on)	31.012146
Row weights	[14, 14, 14, 14, 12, 12, 12, 12]
Number of ones in the matrix	104

42 bits total length, 35 bits data, 7 check bits

Matrix: [1, 2, 4, 8, 16, 32, 64, 7, 11, 19, 35, 67, 13, 21, 37, 69, 25, 41, 73, 49, 81, 97, 14, 22, 38, 70, 26, 42, 74, 50, 82, 98, 28, 44, 76, 52, 84, 100, 56, 88, 104, 112]

Check part of the matrix as bits:

0000100010010110001001010110010110111

0001000100101010010010101010110011

0010001001001100100100110100110110 1

0100010001110001000111000111000111 0

1000011110000001111000000111111000 0

1111100000000001111111110000000000

1111111111111110000000000000000000

No double columns, can be used as SEC code.

Type: Hsia-Code (all columns have odd weight)

Table 9-11 Code Characteristics 35/7

Number of codes with weight 3	0
Number of codes with weight 4	1855
Can be used as SECDED code	
Undetectable double errors (correction on)	0
Undetectable triple errors (correction on)	7420
Percentage of undetectable triple errors (correction on)	64.63415

Table 9-11 Code Characteristics 35/7

Row weights	[16, 16, 16, 16, 16, 16, 16]
Number of ones in the matrix	112

44 bits total length, 36 bits data, 8 check bits

Matrix: [1, 2, 4, 8, 16, 32, 64, 128, 21, 28, 74, 26, 7, 70, 140, 112, 138, 11, 196, 161, 22, 41, 193, 152, 100, 35, 56, 137, 200, 37, 131, 168, 50, 73, 82, 176, 81, 14, 52, 164, 84, 224, 42, 97]

Check part of the matrix as bits:

0000001010110011000110110001000101000

0010010100100010100010000110100011101

0000000100010100111001011001001101111

1101000100000100100100000101110101000

0111001011000101001110010100010000010

1100111000101000100001000000001111000

0011110011001000010000101010010000010

1000100001010110010101100100100000001

No double columns, can be used as SEC code.

Type: Hsiao-Code (all columns have odd weight)

Table 9-12 Code Characteristics 36/8

Number of codes with weight 3	0
Number of codes with weight 4	1143
Can be used as SECDED code	
Undetectable double errors (correction on)	0
Undetectable triple errors (correction on)	4572
Percentage of undetectable triple errors (correction on)	34.521294
Row weights	[14, 14, 14, 16, 14, 16, 14, 14]
Number of ones in the matrix	116

72 bits total length, 64 bits data, 8 check bits

Matrix: [1, 2, 4, 8, 16, 32, 64, 128, 7, 11, 19, 35, 67, 131, 13, 21, 37, 69, 133, 25, 41, 73, 137, 49, 81, 145, 97, 161, 193, 14, 22, 38, 70, 134, 26, 42, 74, 138, 50, 82, 146, 98, 162, 194, 28, 44, 76, 140, 52, 84, 148, 100, 164, 196, 56, 88, 152, 104, 168, 200, 112, 176, 208, 224, 227, 233, 143, 62, 93, 47, 181, 124]

Check part of the matrix as bits:

00000100001000100101100001000100101100010010110010110111111100010
00001000010001001010100010001001010100100101010101011011110010 01
00010000100010010011000100010010011001001001101001101101110101 11
00100001000100011100001000100011100010001110001110001110000110 11
01000010000111100000010000111100000011110000001111110000011111 01
10000011111000000000011111000000000011111111110000000000001111 11
11111100000000000000011111111111111100000000000000000000010110 100
11111111111111111111000000000000000000000000000000000011101 110

No double colums, can be used as SEC code.

Type: Hsiao-Code (all columns have odd weight)

Table 9-13 Code Characteristics 64/8

Number of codes with weight 3	0
Number of codes with weight 4	8414
Undetectable double errors (correction on)	0
Undetectable triple errors (correction on)	33656
Percentage of undetectable triple errors (correction on)	56.431927
Row weights	[28, 26, 28, 28, 26, 28, 26, 26]
Number of ones in the matrix	216

137 bits total length, 128 bits data, 9 check bits

Matrix: [1, 2, 4, 8, 16, 32, 64, 128, 256, 19, 21, 25, 37, 38, 44, 49, 52, 67, 69, 73, 81, 82, 84, 88, 97, 100, 112, 131, 133, 137, 145, 146, 148, 152, 161, 164, 176, 193, 194, 196, 200, 208, 224, 448, 385, 265, 266, 400, 321, 268, 273, 296, 292, 280, 328, 336, 392, 15, 30, 43, 58, 78, 106, 142, 170, 306, 354, 263, 278, 326, 291, 418, 390, 55, 61, 103, 117, 118, 124, 167, 173, 181, 182, 188, 211, 213, 217, 230, 236, 109, 345, 229, 346, 241, 244, 409, 425, 410, 484, 457, 364, 428, 412, 458, 361, 460, 421, 357, 301, 283, 285,

436, 302, 309, 313, 316, 331, 440, 333, 472, 348, 488, 372, 376, 395, 95, 123, 207, 250, 187, 407, 455, 498, 470, 435, 371, 445]

Check part of the matrix as bits:

000000000000000000000000000000001111111111111100000000111111110000000000000010100111111111111111111111111111111111000001111111

000000000000000000011111111111111111100100000000100000011000000110000001111000111111111111111111111111111111111111111111111

000000001111111111000000000011111110000100000110000011000010010000011110000011111111111000111001110100000000101111110111100111010

000111110000000111000000011100000100000000110000001101011100011011111111110001110101101010110010111001111101000111001011001011

111000110001111001000111100100001000001001001010010100010010000110111001111000101111010000100000111011101011010101100101110011111010001110010110010111

100111111000101111010000100000011101110101101101101110111111111100000000010001001000010001000010001000001000011001010101110111111111000000000100010100100101110100111011111110011101011111111101111111100000001

010111010100010010010001001000100000000010010000110010100011100111111111101011101000100100100010111001111011101000101110011110111010100101001001101001

100010001000100000100010000001000000010000000000111111111111111110101010010001000010000010000010000010010001000000011111111111110

111100101111000100111100010010000001100101000000101000000010010011110011100111001110101100100001011111001101010000011110111001111

No double colums, can be used as SEC code.

Type: No Hsiao code (at least one row has even weight

Table 9-14 Code Characteristics 128/9

Number of codes with weight 3	0
Number of codes with weight 4	45921
Undetectable double errors (correction on)	0
Undetectable triple errors (correction on)	183684
Percentage of undetectable triple errors (correction on)	43.81566
Row weights	[62, 50, 62, 62, 64, 64, 62, 62, 62]
Number of ones in the matrix	550

DED Codes

2-Bit-error detection, but no correction possible!

25 bits total length, 20 bits data, 5 check bits

Matrix: [1, 2, 4, 8, 16, 3, 5, 9, 17, 6, 10, 18, 12, 20, 24, 7, 11, 19, 13, 21, 14, 26, 28, 15, 31]

Check part of the matrix as bits:

0001001011001010110 1

0010010101010111 11

0100100110100111 0111

1000111000111001 1011

1111000000111110 0011

No double colums, can be used as SEC code.

Type: No Hsiao code (at least one row has even weight)

Table 9-15 Code Characteristics 20/5

Number of codes with weight 3	80
Number of codes with weight 4	435
Can be used as DED code only	
Percentage of undetectable triple errors	3.4782608
Row weights	[12, 12, 12, 12, 10]
Number of ones in the matrix	58

26 bits total length, 21 bits data, 5 check bits

Matrix: [1, 2, 4, 8, 16, 3, 5, 9, 17, 6, 10, 18, 12, 20, 24, 7, 11, 19, 13, 21, 25, 14, 22, 26, 28, 31]

Check part of the matrix as bits:

0001001011001011011 11

0010010101010110 1111

0100100110100110 1011

1000111000111000 1101

1111000000111111 0001

No double colums, can be used as SEC code.

Type: No Hsiao code (at least one row has even weight

Table 9-16 Code Characteristics 21/5

Number of codes with weight 3	90
Number of codes with weight 4	515
Can be used as DED code only	
Percentage of undetectable triple errors	3.4615386
Row weights	[12, 12, 12, 12, 12]
Number of ones in the matrix	60

27 bits total length, 22 bits data, 5 check bits

Matrix: [1, 2, 4, 8, 16, 3, 5, 9, 17, 6, 10, 18, 12, 20, 24, 7, 11, 19, 13, 21, 25, 14, 22, 26, 15, 23, 27]

Check part of the matrix as bits:

0001001011001011011011

0010010101010101101101

0100100110100110110110

1000111000111000111111

1111000000111111000111

No double colums, can be used as SEC code.

Type: No Hsiao code (at least one row has even weight)

Table 9-17 Code Characteristics 22/5

Number of codes with weight 3	101
Number of codes with weight 4	606
Can be used as DED code only	
Percentage of undetectable triple errors	3.4529915
Row weights	[14, 14, 12, 12, 12]
Number of ones in the matrix	64

30 bits total length, 24 bits data, 6 check bits

Matrix: [1, 2, 4, 8, 16, 32, 3, 5, 9, 17, 33, 6, 10, 18, 34, 12, 20, 36, 24, 40, 7, 11, 19, 35, 13, 49, 22, 42, 52, 56]

Check part of the matrix as bits:

000010001001010001010111

000100010010100010011011

001000100100110100100101

010001000111001000101010

100001111000001111001100

111110000000001111110000

No double colums, can be used as SEC code.

Type: No Hsiao code (at least one row has even weight)

Table 9-18 Code Characteristics 24/6

Number of codes with weight 3	75
Number of codes with weight 4	458
Can be used as DED code only	
Percentage of undetectable triple errors	1.8472906
Row weights	[12, 12, 10, 10, 10, 10]
Number of ones in the matrix	64

71 bits total length, 64 bits data, 7 check bits

Matrix: [1, 2, 4, 8, 16, 32, 64, 3, 5, 9, 17, 33, 65, 6, 10, 18, 34, 66, 12, 20, 36, 68, 24, 40, 72, 48, 80, 96, 7, 11, 19, 35, 67, 13, 21, 37, 69, 25, 41, 73, 49, 81, 97, 14, 22, 38, 70, 26, 42, 74, 50, 82, 98, 28, 44, 76, 52, 84, 100, 56, 88, 104, 112, 15, 23, 43, 51, 101, 89, 102, 90]

Check part of the matrix as bits:

00000100001000100101100001000100101100010010110010110111000011111

00001000010001001010100010001001010100100101010101011011001110101

00010000100010010011000100010010011001001001101001101101010101010

00100001000100011100001000100011100010001110001110001110101001010

01000010000111100000010000111100000011110000001111110000110010101

10000011111000000000011111000000000011111111110000000000111100 11
11111100000000000000011111111111111100000000000000000000011111100

No double colums, can be used as SEC code.

Type: No Hsiao code (at least one row has even weight)

Table 9-19 Code Characteristics 64/7

Number of codes with weight 3	505
Number of codes with weight 4	7933
Can be used as DED code only	
Percentage of undetectable triple errors	0.88356227
Row weights	[28, 28, 26, 26, 26, 26, 26]
Number of ones in the matrix	186

136 bits total length, 128 bits data, 8 check bits

Matrix: [1, 2, 4, 8, 16, 32, 64, 128, 3, 5, 9, 17, 33, 65, 129, 6, 10, 18, 34, 66, 130, 12, 20, 36, 68, 132, 24, 40, 72, 136, 48, 80, 144, 96, 160, 192, 7, 11, 19, 35, 67, 131, 13, 21, 37, 69, 133, 25, 41, 73, 137, 49, 81, 145, 97, 161, 193, 14, 22, 38, 70, 134, 26, 42, 74, 138, 50, 82, 146, 98, 162, 194, 28, 44, 76, 140, 52, 84, 148, 100, 164, 196, 56, 88, 152, 104, 168, 200, 112, 176, 208, 224, 15, 216, 202, 75, 150, 92, 89, 149, 204, 60, 228, 153, 108, 99, 166, 172, 135, 169, 46, 51, 198, 116, 106, 86, 240, 78, 209, 177, 27, 141, 225, 232, 147, 43, 212, 163, 39, 45, 90, 210, 57, 23, 29, 165]

Check part of the matrix as bits:

00000010000010000100010010110000010000100010010110000100010010110001 00101100101101110110100110110011110010001011011110110001000 1

00000100000100001000100101010000100001000100101010001000100101010010 01010101011011011101101010110000001111111000110010001100 00

00001000001000010001001001100001000010001001001100010001001001100100 10011010011011010000000011011110111011010010011010111001001

00010000010000100010001110000010000100010001110000100010001110001000 11000111000111001001111010100000001010110111000101000111110

00100000100001000011110000000100001000011110000001000011110000001110 00000111111000011110110110110010110001001001101010001101010

01000001000001111100000000001000001111100000000001111100000000001111 11111000000000010001101111010111010110101000100001011000111

1000000111111000000000000000011111100000000000000011111111111111000000000000000000010111000000000110101110110100100011011011010011111111000000000000000000000000000011111111111111111111111000000000000000000000000000000000000000000000100100110001010011010000001111101101110011111

No double colums, can be used as SEC code.

Type: No Hsiao code (at least one row has even weight)

Table 9-20 Code Characteristics 64/7

Number of codes with weight 3	1812
Number of codes with weight 4	55286
Can be used as DED code only	
Percentage of undetectable triple errors	0.44190812
Row weights	[52, 50, 52, 52, 50, 50, 50, 52]
Number of ones in the matrix	408

9.5.2 Address Error Detection

Address error detection has to capture several memory faults that cause multiple or all bits of a word to fail. Such faults could be address decoder stuck-at-faults, broken word-lines, bridged word-lines etc. Because of this origin an ECC detection would not cover these cases because the words read back from memory would have perfectly matched check bits despite corrupted word selects.

In this version of MBIST/ECC the SRAMs will detect any of the faulty cases.

.

9.6 Implementation Section

All Memory Controller register regions which are not allocated in the table below are not implemented. Attempted accesses to regions which are not implemented will return a bus error.

9.6.1 Memory Control Register Implementation

9.6.1.1 MEMTEST Implementation

The memory test register MEMTEST contains enable bits for the various Memory Controllers.

Memory control registers described in this chapter are accessible only when the Memory Controller is enabled (MTU_MEMTEST.MEMxEN = 1).The only exceptions to this are registers ECCDand ETRR (for each memory) which are available at all times to permit easy runtime access to ECC features.

Note:

2. *When a Memory Controller is enabled, functional access to the memory is temporarily unavailable. When a memory is being tested it is essential to prevent an attempted functional access to that memory (e.g. While testing a CPU's local memories, the CPU should be in an idle state, or executing from other memory). Attempted functional accesses to memories when MEMxEN=1 may result in unexpected behaviour.*

3. *Correct execution of a Memory Controller operation (e.g. MBIST or initialization sequence) requires that both the MTU and the module containing the memory-under-test are both operational for the duration of the sequence. A module reset of the MTU or the module containing the memory-under-test, during execution of such an MBIST sequence may result in unexpected behaviour.*

MTU_MEMTEST0
Memory MBISTEnable Register 0 (F006 0010$_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GTM 1AEN	**GTM M1EN**	**GTM M0EN**	**GTM FEN**	**CPU 0DS 2EN**	**FSI0 EN**	**Res**	**Res**	**Res**	**ETH EN**	**CPU 2DS 2EN**	**CPU 1DS 2EN**	**CPU 0DT EN**	**Res**	**CPU 0PT EN**	**CPU 0PS EN**
rwh	rwh	rwh	rwh	rwh	rwh	r	r	r	rwh	rwh	rwh	rwh	r	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	**CPU 0DS EN**	**MMC DSEN**	**LM-UEN**	**CPU 1PT EN**	**Res**	**CPU 1PS EN**	**CPU 1DT EN**	**Res**	**CPU 1DS EN**	**CPU 2PT EN**	**Res**	**CPU 2PS EN**	**CPU 2DT EN**	**Res**	**CPU 2DS EN**
r	rwh	rwh	rwh	rwh	r	rwh	rwh	r	rwh	rwh	r	rwh	rwh	r	rwh

Field	Bits	Type	Description
CPU2XEN	[5:0]	r	**Reserved in this product**
CPU1XEN	[11:6]	r	**Reserved in this product**
LMUEN	12	r	**Reserved in this product**
MMCDSEN	13	rwh	**MINI MCDS TRAM MBIST Controller Memory Enable** 0_B MBIST controller is disabled 1_B MBIST controller is enabled
CPU0DSEN	14	rwh	**CPU0 DSPR MBIST Controller Memory Enable** 0_B MBIST controller is disabled 1_B MBIST controller is enabled
Res	15	r	**Reserved in this product**
CPU0PSEN	16	rwh	**CPU0 PSPR MBIST Controller Memory Enable** 0_B MBIST controller is disabled 1_B MBIST controller is enabled
CPU0PTEN	17	rwh	**CPU0 PTAG MBIST Controller Memory Enable** 0_B MBIST controller is disabled 1_B MBIST controller is enabled (subject to restrictions)
Res	18	r	**Reserved**
CPU0DTEN	19	r	**Reserved in this product**
CPUXDS2EN	[21:20]	r	**Reserved in this product**
ETHEN	22	rwh	**ETHERMAC MBIST Controller Memory Enable** 0_B MBIST controller is disabled 1_B MBIST controller is enabled
Res	[25:23]	r	**Reserved**
FSI0EN	26	rwh	**FSI0 MBIST Controller Memory Enable** 0_B MBIST controller is disabled 1_B MBIST controller is enabled
CPU0DS2EN	27	rwh	**CPU0 TC1.6E DSPR2 MBIST Controller Memory Enable (TC23x Only)** 0_B MBIST controller is disabled 1_B MBIST controller is enabled (subject to restrictions)
GTMFEN	28	r	**Reserved in this product**
GTMM0EN	29	r	**Reserved in this product**

Field	Bits	Type	Description
GTMM1EN	30	r	**Reserved in this product**
GTM1AEN	31	r	**Reserved in this product**

MTU_MEMTEST1
Memory MBISTEnable Register 1 (F006 0014$_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EMEMU1EN	**EMEMU0EN**	**EMEML15EN**	**EMEML14EN**	**EMEML13EN**	**EMEML12EN**	**EMEML11EN**	**EMEML10EN**	**EMEML9EN**	**EMEML8EN**	**EMEML7EN**	**EMEML6EN**	**EMEML5EN**	**EMEML4EN**	**EMEML3EN**	**EMEML2EN**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EMEML1EN	**EMEML0EN**	**MCDSEN**	**STBY0EN**	**ERAY1MEN**	**ERAY1TEN**	**ERAY1OEN**	**ERAY0MEN**	**ERAY0TEN**	**ERAY0OEN**	**MCAN1EN**	**MCAN0EN**	**Res**	**PSI5EN**	**GTM2EN**	**GTM1BEN**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	r	rwh	rwh	rwh

Field	Bits	Type	Description
GTM1BEN	0	r	**Reserved in this product**
GTM2EN	1	r	**Reserved in this product**
PSI5EN	2	r	**Reserved in this product**
Res	3	r	**Reserved**
MCAN0EN	4	rwh	**MultiCAN0 Controller Memory Enable** 0_B MBIST controller is disabled 1_B MBIST controller is enabled
MCAN1EN	5	rwh	**MultiCAN1 Controller Memory Enable** 0_B MBIST controller is disabled 1_B MBIST controller is enabled
ERAY0OEN	6	rwh	**ERAY0 OBF Controller Memory Enable** 0_B MBIST controller is disabled 1_B MBIST controller is enabled
ERAY0TEN	7	rwh	**ERAY0 TBF Controller Memory Enable** 0_B MBIST controller is disabled 1_B MBIST controller is enabled

Field	Bits	Type	Description
ERAY0MEN	8	rwh	**ERAY0 MBF Controller Memory Enable** 0_B MBIST controller is disabled 1_B MBIST controller is enabled
ERAY1XEN	[11:9]	r	**Reserved in this product**
STBY1EN	12	r	**Reserved in this product**
MCDSEN	13	rwh	**MCDS Controller Memory Enable (ED only)** 0_B MBIST controller is disabled 1_B MBIST controller is enabled
EMEMLxEN (x = 0-3)	x+14	rwh	**EMEM Lower x (TCM) MBIST Controller Memory Test Enable (ED only)** 0_B MBIST controller is disabled 1_B MBIST controller is enabled
EMEMLXEN	[29:18]	r	**Reserved in this product**
EMEMUXEN	[31:30]	r	**Reserved in this product**

MTU_MEMTEST2
Memory MBISTEnable Register 2 (F006 0018$_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								FFT1EN	FFT0EN	XTM1EN	XTM0EN	DMAEN	STBY2EN	CIF2EN	CIF1EN
r								rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAMEN	CIF0EN	EMEMU15EN	EMEMU14EN	EMEMU13EN	EMEMU12EN	EMEMU11EN	EMEMU10EN	EMEMU9EN	EMEMU8EN	EMEMU7EN	EMEMU6EN	EMEMU5EN	EMEMU4EN	EMEMU3EN	EMEMU2EN
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
EMEMUxEN	[13:0]	r	**Reserved in this product**
CIF0EN	14	r	**Reserved in this product**

Field	Bits	Type	Description
DAMEN	15	rwh	**DAM Memory Enable** 0_B MBIST controller is disabled 1_B MBIST controller is enabled
CIFxEN	[17:16]	r	**Reserved in this product**
STBY2EN	18	r	**Reserved in this product**
DMAEN	19	r	**Reserved in this product**
XTM0EN	20	rwh	**EMEM XTM0 Controller Memory Enable (ED only)** 0_B MBIST controller is disabled 1_B MBIST controller is enabled (subject to restrictions)
XTM1EN	21	rwh	**EMEM XTM1 Controller Memory Enable (ED only)** 0_B MBIST controller is disabled 1_B MBIST controller is enabled (subject to restrictions)
FFT0EN	22	rwh	**FFT0 Memory Controller Memory Enable (ADAS Product only)** 0_B MBIST controller is disabled 1_B MBIST controller is enabled (subject to restrictions)
FFT1EN	23	rwh	**FFT1 Memory Controller Memory Enable (ADAS Product only)** 0_B MBIST controller is disabled 1_B MBIST controller is enabled (subject to restrictions)
Res	[31:24]	r	**Reserved**

9.6.1.2 MEMMAP Implementation

The Memory Mapping Enable register MEMMAP has configurable control bits to select memory-mapped test mode for each CPU memory.

Cache and Scratchpad memories are physically implemented as a single RAM, but this register function assumes two separate logical RAM partitions. In this register additional bits CPUxDCMAP and CPUxPCMAP are defined. These control the Cache partitions of the RAMs for Data Side and Program side respectively. Since cache content and tags of a cache must be simultaneously switched from memory mapped to cache functional mode, the control bits are mirrored and only one bit is writeable for each cache. The bits corresponding to the tag memories of the same cache will always take the same value as that written to the main Cache Memory control bit. This linkage is product specific.

MTU_MEMMAP
Memory Mapping Enable Register (F006 001C$_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res												CPU 0DT MAP	CPU 0DC MAP	CPU 0PT MAP	Res
r												rh	rwh	rh	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPU 0PC MAP	Res			CPU 1PT MAP	CPU 1PC MAP	Res	CPU 1DT MAP	CPU 1DC MAP	Res	CPU 2PT MAP	CPU 2PC MAP	Res	CPU 2DT MAP	CPU 2DC MAP	Res
rwh	r	r	r	rh	rwh	r	rh	rwh	r	rh	rwh	r	rh	rwh	r

Field	Bits	Type	Description
Res	0	r	**Reserved** Not used
CPU2DxMAP	[2:1]	r	**Reserved in this product** Not used
Res	3	r	**Reserved** Not used
CPU2PxMAP	[5:4]	r	**Reserved in this product** Not used

Field	Bits	Type	Description
Res	6	r	**Reserved** Not used
CPU1DxMAP	[8:7]	r	**Reserved in this product** Not used
Res	9	r	**Reserved** Not used
CPU1PxMAP	[11:10]	r	**Reserved in this product** Not used in this product.
Res	[14:12]	r	**Reserved** Not used
CPU0PCMAP	15	rwh	**CPU0 PCACHE Mapping** 0_B Normal cache function 1_B Memory-mapped
Res	16	r	**Reserved** Not used
CPU0PTMAP	17	rh	**CPU0 PTAG Mapping** 0_B Normal cache function 1_B Memory-mapped **Read only. Mirrors the state of CPU0PCEN.** CPU0 cache memories may only be mapped simultaneously.
CPU0DxMAP	[19:18]	r	**Reserved in this product** Not used in this product.
Res	[31:20]	r	**Reserved** Not used

9.6.1.3 MEMSTAT Implementation

The Memory Status registers MEMSTATx have an implemented bit for each security-relevant RAM.

Cache and Scratchpad memories are physically implemented as a single RAM with a single MBIST, so bits CPUxDSAIUand CPUxPSAIU give the status of the partial initialization of the cache partitions for Data Side and Program side respectively. Since cache content and tags of a cache may take different lengths of time to erase, both are implemented as separate status bits.

MTU_MEMSTAT0
Memory Status Register 0 (F006 0038$_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res				CPU 0DS 2AIU	FSI0 AIU	HSM RAIU	HSM TAIU	HSM CAIU	Res	CPU 2DS 2AIU	CPU 1DS 2AIU	CPU 0DT AIU	Res	CPU 0PT AIU	CPU 0PS AIU
r				rh	rh	rh	rh	rh	r	rh	rh	rh	r	r	rh

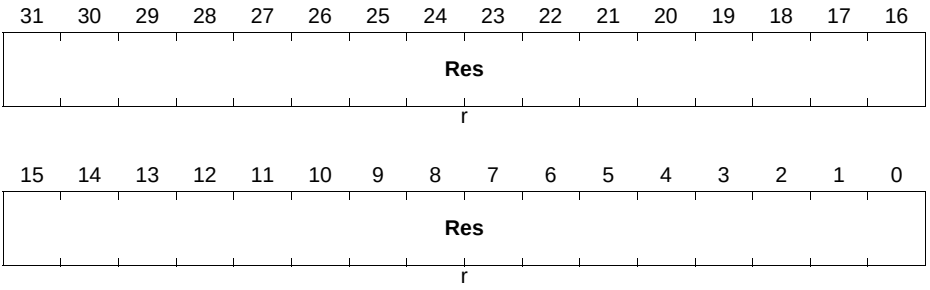
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	CPU 0DS AIU	Res		CPU 1PT AIU	Res	CPU 1PS AIU	CPU 1DT AIU	Res	CPU 1DS AIU	CPU 2PT AIU	Res	CPU 2PS AIU	CPU 2DT AIU	Res	CPU 2DS AIU
r	rh	r	r	rh	r	rh	rh	r	rh	rh	r	rh	rh	r	rh

Field	Bits	Type	Description
CPU2DSAIU	0	r	**Reserved in this product**
CPU2DTAIU	2	r	**Reserved in this product**
CPU2PSAIU	3	r	**Reserved in this product**
CPU2PTAIU	5	r	**Reserved in this product**
CPU1DSAIU	6	r	**Reserved in this product**
CPU1DTAIU	8	r	**Reserved in this product**
CPU1PSAIU	9	r	**Reserved in this product**
CPU1PTAIU	11	r	**Reserved in this product**
CPU0DSAIU	14	r	**Reserved in this product**

Field	Bits	Type	Description
CPU0PSAIU	16	rh	**CPU0 PCACHE Partial AutoInitialize Underway** 0_B MBIST not running autoinitialize 1_B MBIST running autoinitialize This bit indicates whether an automatic data initialization has been triggered by a change of state of MEMTEST.MEMxEN or MEMxMAP but that the intialization sequence has not yet completed.
CPU0PTAIU	17	rh	**CPU0 PTAG MBIST AutoInitialize Underway** 0_B MBIST not running autoinitialize 1_B MBIST running autoinitialize This bit indicates whether an automatic data initialization has been triggered by a change of state of MEMTEST.MEMxEN or MEMxMAP but that the intialization sequence has not yet completed.
CPU0DxAIU	19	r	**Reserved in this product**
CPU1DS2AIU	20	r	**Reserved in this product**
CPU2DS2AIU	21	r	**Reserved in this product**
HSMCAIU	23	r	**Reserved in this product** Not used in this product.
HSMTAIU	24	r	**Reserved in this product** Not used in this product.
HSMRAIU	25	r	**Reserved in this product** Not used in this product.
FSI0AIU	26	rh	**FSI0 MBIST AutoInitialize Underway** 0_B MBIST not running autoinitialize 1_B MBIST running autoinitialize This bit indicates whether an automatic data initialization has been triggered by a change of state of MEMTEST.MEMxEN or MEMxMAP but that the intialization sequence has not yet completed.
CPU0DS2AIU	27	rh	**CPU0 DCACHE Partial AutoInitialize Underway (TC23x Second DS Bank)** 0_B MBIST not running autoinitialize 1_B MBIST running autoinitialize This bit indicates whether an automatic data initialization has been triggered by a change of state of MEMTEST.MEMxEN or MEMxMAP but that the intialization sequence has not yet completed.

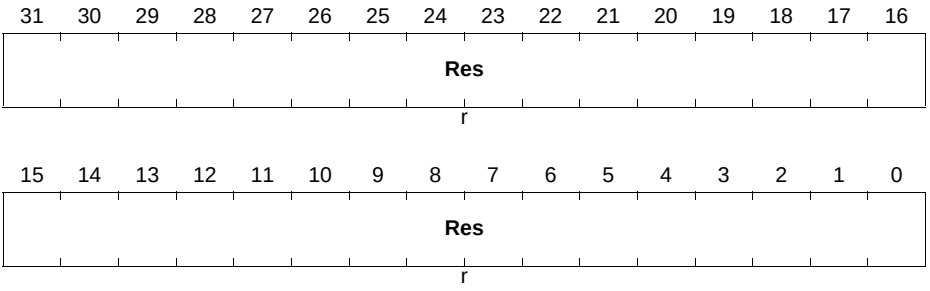
Field	Bits	Type	Description
0	31,30, 29,28, 22,18, 15,13, 12,10, 7,4,1	r	**Reserved** Not used

MTU_MEMSTAT1
Memory Status Register 1 **(F006 003C$_H$)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
Res	[31:0]	r	**Reserved** Not used

MTU_MEMSTAT2
Memory Status Register 2 **(F006 0040$_H$)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
Res	[31:0]	r	**Reserved** Not used

9.6.1.4 Memory Controller Instances

Table 9-21 Address Regions - Memory Controller Register

No.	Module	Base Address	End Address	Note
0	Reserved in this product	F006 1000$_H$	F006 10FF$_H$	-
1	Reserved	F006 1100$_H$	F006 11FF$_H$	-
2	Reserved in this product	F006 1200$_H$	F006 12FF$_H$	-
3	Reserved in this product	F006 1300$_H$	F006 13FF$_H$	-
4	Reserved	F006 1400$_H$	F006 14FF$_H$	-
5	Reserved in this product	F006 1500$_H$	F006 15FF$_H$	-
6	Reserved in this product	F006 1600$_H$	F006 16FF$_H$	-
7	Reserved	F006 1700$_H$	F006 17FF$_H$	-
8	Reserved in this product	F006 1800$_H$	F006 18FF$_H$	-
9	Reserved in this product	F006 1900$_H$	F006 19FF$_H$	-
10	Reserved	F006 1A00$_H$	F006 1AFF$_H$	-
11	Reserved in this product	F006 1B00$_H$	F006 1BFF$_H$	-
12	Reserved	F006 1C00$_H$	F006 1CFF$_H$	-
13	MC_MINI_MCDS_TRAM	F006 1D00$_H$	F006 1DFF$_H$	-
14	MC_CPU0_DSPR	F006 1E00$_H$	F006 1EFF$_H$	-
16	MC_CPU0_PSPR	F006 2000$_H$	F006 20FF$_H$	-
17	MC_CPU0_PTAG	F006 2100$_H$	F006 21FF$_H$	-
18	Reserved	F006 2200$_H$	F006 22FF$_H$	-
19	Reserved in this product	F006 2300$_H$	F006 23FF$_H$	-
20	Reserved in this product	F006 2400$_H$	F006 24FF$_H$	-
21	Reserved in this product	F006 2500$_H$	F006 25FF$_H$	-
22	MC_ETHERMAC	F006 2600$_H$	F006 26FF$_H$	-

Table 9-21 Address Regions - Memory Controller Register

No.	Module	Base Address	End Address	Note
23	Reserved in this product	F006 2700$_H$	F006 27FF$_H$	-
24	Reserved in this product	F006 2800$_H$	F006 28FF$_H$	-
25	Reserved in this product	F006 2900$_H$	F006 29FF$_H$	-
26	MC_FSI0	F006 2A00$_H$	F006 2AFF$_H$	-
27	MC_CPU0_DSPR2	F006 2B00$_H$	F006 2BFF$_H$	-
27	Reserved	F006 2B00$_H$	F006 2BFF$_H$	-
28	Reserved in this product	F006 2C00$_H$	F006 2CFF$_H$	-
29	Reserved in this product	F006 2D00$_H$	F006 2DFF$_H$	-
30	Reserved in this product	F006 2E00$_H$	F006 2EFF$_H$	-
31	Reserved in this product	F006 2F00$_H$	F006 2FFF$_H$	-
32	Reserved in this product	F006 3000$_H$	F006 30FF$_H$	-
33	Reserved in this product	F006 3100$_H$	F006 31FF$_H$	-
34	Reserved in this product	F006 3200$_H$	F006 32FF$_H$	-
35	Reserved	F006 3300$_H$	F006 33FF$_H$	-
36	MC_MCAN0	F006 3400$_H$	F006 34FF$_H$	-
37	MC_MCAN1	F006 3500$_H$	F006 35FF$_H$	-
38	MC_ERAY0 OBF	F006 3600$_H$	F006 36FF$_H$	-
39	MC_ERAY0 IBF_TBF	F006 3700$_H$	F006 37FF$_H$	-
40	MC_ERAY0 MBF	F006 3800$_H$	F006 38FF$_H$	-
41	Reserved in this product	F006 3900$_H$	F006 39FF$_H$	-
42	Reserved in this product	F006 3A00$_H$	F006 3AFF$_H$	-
43	Reserved in this product	F006 3B00$_H$	F006 3BFF$_H$	-
44	Reserved in this product	F006 3C00$_H$	F006 3CFF$_H$	-
45	MC_MCDS (ED only)	F006 3D00$_H$	F006 3DFF$_H$	-
-	Reserved in this product	F006 3E00$_H$	F006 5DFF$_H$	-
46	MC_EMEM LOWER 0 (ED only)	F006 3E00$_H$	F006 3EFF$_H$	-
47	MC_EMEM LOWER 1 (ED only)	F006 3F00$_H$	F006 3FFF$_H$	-
48	MC_EMEM LOWER 2 (ED only)	F006 4000$_H$	F006 40FF$_H$	-
49	MC_EMEM LOWER 3 (ED only)	F006 4100$_H$	F006 41FF$_H$	-
-	Reserved in this product	F006 4200$_H$	F006 45FF$_H$	-

Table 9-21 Address Regions - Memory Controller Register

No.	Module	Base Address	End Address	Note
-	Reserved in this product	F006 4600$_H$	F006 5DFF$_H$	-
78	Reserved in this product	F006 5E00$_H$	F006 5EFF$_H$	-
79	MC_DAM	F006 5F00$_H$	F006 5FFF$_H$	-
80	Reserved in this product	F006 6000$_H$	F006 60FF$_H$	-
81	Reserved in this product	F006 6100$_H$	F006 61FF$_H$	-
82	Reserved in this product	F006 6200$_H$	F006 62FF$_H$	-
83	Reserved in this product	F006 6300$_H$	F006 63FF$_H$	-
84	MC_EMEM XTM0 (ED only)	F006 6400$_H$	F006 64FF$_H$	-
85	MC_EMEM XTM1 (ED only)	F006 6500$_H$	F006 65FF$_H$	-
84	Reserved in this product	F006 6400$_H$	F006 64FF$_H$	-
85	Reserved in this product	F006 6500$_H$	F006 65FF$_H$	-
86	MC_FFT0 (ADAS Product only)	F006 6600$_H$	F006 66FF$_H$	-
87	MC_FFT1 (ADAS Product only)	F006 6700$_H$	F006 67FF$_H$	-
	Reserved.			All other unassigned addresses up to F006 FFFF$_H$

The system SRAMs do not all have the same configuration. **Chapter 9-22** shows the instance-specific configurations of the MBIST modules.

The ECC values for all SRAMs are computed only out of the data information*.

Table 9-22 Configurations - Memory Controllers

No.	Module	Error Addr Buffer Depth	ECC type	ECC granularity	Mux Factor
13	MINI_MCDS_TRAM	1	SECDED	1	4
14	MC_CPU0_TC16E_DSPR	5	SECDED	2	16
15	MC_CPU0_TC16E_DSPR2	5	SECDED	2	16
16	MC_CPU0_TC16E_PSPR	3	SECDED	1	8

Table 9-22 Configurations - Memory Controllers

No.	Module	Error Addr Buffer Depth	ECC type	ECC granularity	Mux Factor
17	MC_CPU0_TC16E_PTAG	2	DED	2	4
20	MC_FFT0	5	SECDED	1	4
21	MC_FFT1	1	SECDED	1	4
22	MC_ETHERMAC	1	SECDED	1	4
26	MC_FSI0	1	SECDED	1	8
36	MC_MCAN0	1	SECDED	1	4
37	MC_MCAN1	1	SECDED	1	4
38	MC_ERAY0 OBF	1	SECDED	1	4
39	MC_ERAY0 IBF_TBF	1	SECDED	1	4
40	MC_ERAY0 MBF	1	SECDED	1	4
46	MC_EMEM LOWER 0 (ED only)	1	SECDED	1	4
47	MC_EMEM LOWER 1 (ED only)	1	SECDED	1	4
48	MC_EMEM LOWER 2 (ED only)	1	SECDED	1	4
49	MC_EMEM LOWER 3 (ED only)	1	SECDED	1	4
79	MC_DAM	1	SECDED	1	8
84	MC_EMEM XTM0 (ED only)	1	SECDED	1	4
85	MC_EMEM XTM1 (ED only)	1	SECDED	1	4
86	MC_FFT0 (ADAS Product only)	1	SECDED	1	-
87	MC_FFT1 (ADAS Product only)	1	SECDED	1	-

10 Safety Management Unit (SMU)

10.1 Introduction

The SMU is a central and modular component of the safety architecture providing a generic interface to manage the behavior of the microcontroller under the presence of faults. The SMU centralizes all the alarm signals related to the different hardware and software-based safety mechanisms. Each alarm can be individually configured to trigger internal actions and/or notify externally the presence of faults via a fault signaling protocol. The default fault signaling protocol is a bi-stable Error Pin. The severity of each alarm shall be configured according to the needs of the safety application(s): per default every alarm is disabled with the exception of the watchdog timeout alarms. For debug and diagnosis purposes the alarm events are logged, the logged information is resilient to any system or application reset. The SMU also implements some housekeeping functions related to the management and test of dedicated safety mechanisms. A special test mode is available to test the SMU itself enabling to detect latent faults as required by the ISO 26262 safety standard. In addition to the generic register access protection, the SMU implements an independent configuration locking mechanism.

The SMU in combination with the embedded safety mechanisms enable to detect and report more than 99% of the critical failure modes of the microcontroller within the fault tolerance time interval. The timing characteristics of the fault tolerance time interval can be configured in the SMU.

Figure 10-1 “SMU Interfaces” on Page 10-2 gives an overview of the SMU interfaces.

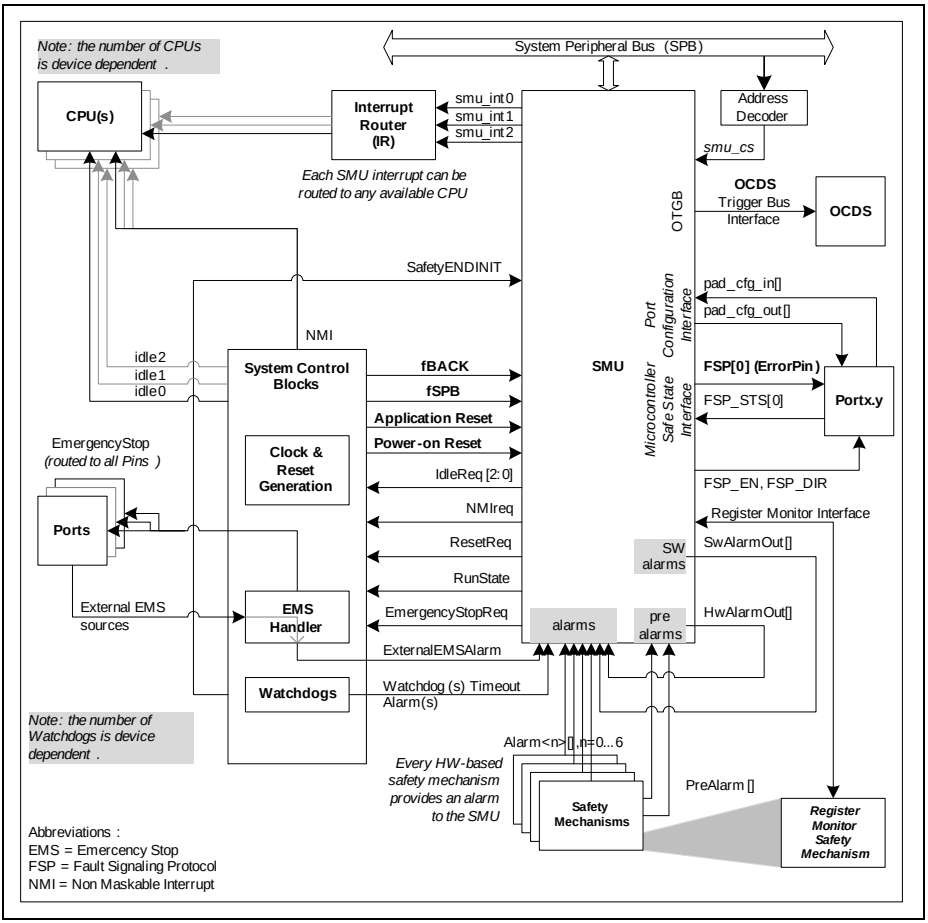


Figure 10-1 SMU Interfaces

10.2 Features

The SMU implements the following features:

- Collects every alarm (error notification) information from every safety mechanism:
  - CPU Lockstep
  - SRAM single bit error correction
  - SRAM uncorrectable error
  - SRAM Monitor: Address Buffer overflow
  - SRAM addressing fault
  - Program FLASH single bit error correction
  - Program FLASH double bit error correction
  - Program FLASH arbitrary multiple-bit uncorrectable error
  - Program FLASH Monitor: address buffer full
  - Program FLASH addressing fault
  - Processor interconnect (SRI) safety mechanisms
  - Safety Flip-Flop
  - Clock monitors
  - System PLL & Flexray-PLL loss of lock
  - Input/Output Monitoring Module
  - Interrupt Router and Interrupt Control Units hardware monitor
  - Voltage Monitor for internal voltage regulators
  - Error reporting from Shared Resource Interconnect and System Peripheral Bus
  - Die Temperature Sensor
- Alarm flags are stored in a debug register that is only reset by the Power-on reset, to enable fault diagnosis and possible recovery.
- An alarm emulation facility is provided to enable software-based diagnostics to post an alarm condition with the same properties as the hardware alarms.
- Implements the access protection and SafetyEndinit modes to protect configuration registers.
- Implements a Fault Signaling Protocol (FSP) reporting internal faults to the external environment. The FSP can be configured using the following modes:
  - Bi-stable single pin output (push-pull active low configuration using FSP[0]), also called ErrorPin.
  - Single-bit timed protocol using FSP[0]
- The FSP value driven by the microcontroller can be observed via an internal status flag
  - Additionally a monitor is available to check the timing and state properties of the FSP protocol when a fault is reported.
- After power-on reset the FSP is disabled, software needs to connect the FSP to the assigned port. The port is configured as GPIO.
- Each individual alarm can be configured to activate the fault signaling protocol and/or one of the following internal actions:

  - generate an Interrupt request to any of the CPUs, concurrent interrupts to several CPUs can be configured
  - generate a NMI request to the System Control Unit
  - generate a reset request to the System Control Unit leading to a system or application reset
  - activate the pad emergency stop signal controlling the safe state of output pads
  - generate a CPU idle request
- After reset every alarm (except watchdog timeout) is disabled.
- A lock mechanism is available to protect the SMU configuration in addition to the standard access protection
- Implements internal watchdog(s) timeout pre-warning function.
- Implements an internal watchdog called recovery timer to monitor the execution of critical software error handlers. The watchdog is started automatically by hardware according to configurable alarm events.

10.3 Functional Overview

The SMU collects all the error flags (called alarms in this document) from all the hardware monitors (safety mechanisms) defined by the safety concept. The section **“Alarm Mapping” on Page 10-14** specifies the alarm interface and classifies them into alarm groups. The alarm groups do not define any hierarchy but only a logical mapping used to map input alarm signals to internal configuration registers. The section **“Alarm Handling” on Page 10-41** describes the configuration options that can be specified. The configuration options specify the behavior of the SMU when an alarm event is detected. The alarm event can trigger an internal action and/or the activation of the Error Pin that indicates the presence of a dangerous fault to the external environment. The section **“SMU Control Interface” on Page 10-48** specifies how the SMU can be controlled by software and the dependencies with the hardware operation. The section **“Fault Signaling Protocol (FSP)” on Page 10-52** describes the properties of the external fault signaling protocol defining the timing and logical properties of the Error Pin(s).

The SMU implements the technical safety requirements

10.4 Functional Description

10.4.1 Reset Types

The SMU requires multiple reset types for its operation. The reset types are fully specified in the System Control Units. The reset types that are required by the SMU are:

- Power-on Reset.
- Debug Reset.
- Application Reset.

Additionally the SMU supports the **module reset** feature. The module reset acts as an application reset, it is fully controlled by software using the **SMU_KRST0**, **SMU_KRST1**, **SMU_KRSTCLR** registers. The module reset only affects the SMU kernel and not the System Peripheral Bus (SPB) bus interface logic.

Table 10-1 “Effect of Reset Types to SMU functionality” on Page 10-6 specifies the scope of each reset type to the SMU functions (a function includes the control and configuration registers and the related logic).

Table 10-1 Effect of Reset Types to SMU functionality

SMU Function	Module Reset	Application Reset	Debug Reset	System Reset	Power-on Reset
SMU FSP Function **Chapter 10.4.8**	Not Affected	Not Affected	Not Affected	Not Affected	Reset
SMU SSM Function **Chapter 10.4.7**	Not Affected	Not Affected	Not Affected	Not Affected	Reset
SMU Debug Function **Chapter 10.4.9**	Not Affected	Not Affected	Reset	Not Affected	Reset
SMU Alarm Debug Registers **Chapter 10.5.6**	Not Affected	Not Affected	Not Affected	Not Affected	Reset
SMU_PCTL.PCS Register Field **SMU_PCTL**	Not Affected	Not Affected	Not Affected	Not Affected	Reset

Table 10-1 Effect of Reset Types to SMU functionality (cont'd)

SMU Function	Module Reset	Application Reset	Debug Reset	System Reset	Power-on Reset
SMU SPB BPI	Not Affected	Reset	Not Affected	Reset	Reset
SMU Other Functions	Reset	Reset	Not Affected	Reset	Reset

10.4.2 Interfaces Overview

This section describes the main interface signals between the SMU and the other modules.

10.4.2.1 Interfaces to SCU

Internal actions resulting from an alarm event that interface to the System Control Unit. The interface signals are:

- Emergency Stop Request
- Reset Request
- Idle Request
- NMI Request

10.4.2.2 Interfaces to the Interrupt Router

Internal actions resulting from an alarm event that interface to the Interrupt Router. The interface signals are:

- SMU Interrupt Request 0
- SMU Interrupt Request 1
- SMU Interrupt Request 2

The mapping of SMU Interrupt Requests to the Interrupt Router (IR) interrupt nodes can be found in the IR chapter.

The **SMU_AGC**.IGCS<i>, i={0,1,2} register fields provide the software interface to control how the SMU triggers interrupt requests to the interrupt router.

Each **SMU_AGC**.IGCS<i> is a 3-bits bit-field: **SMU_AGC**.IGCS<i>[2:0] that defines a direct relationship to the 3 SMU interrupt requests:

- **SMU_AGC**.IGCS<i>[0] shall be set to ‘1’ to trigger SMU Interrupt Request 0
- **SMU_AGC**.IGCS<i>[1] shall be set to ‘1’ to trigger SMU Interrupt Request 1
- **SMU_AGC**.IGCS<i>[2] shall be set to ‘1’ to trigger SMU Interrupt Request 2

The usage of the three **SMU_AGC**.IGCS<i> bit-fields is defined in **“Alarm Configuration” on Page 10-41**; where each **SMU_AGC**.IGCS<i> can be selected as an internal action.

10.4.2.3 Interface to the Ports (Error Pin)

The generic port structure is presented in presented in **Figure 10-2 “Generic Port Structure” on Page 10-9**.

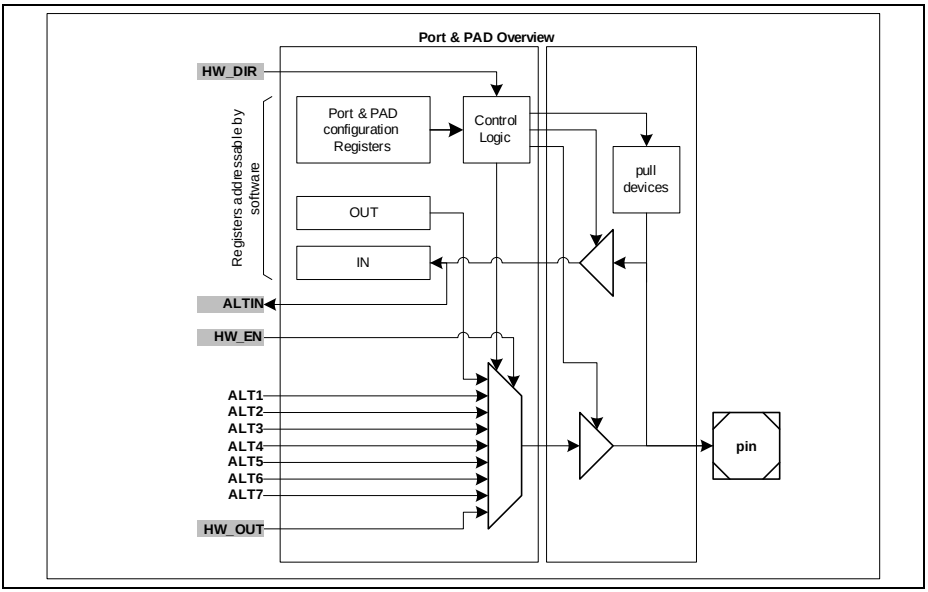


Figure 10-2 Generic Port Structure

The port can be connected to peripheral functions via the ALTx inputs. This is the default state of the port after power-on reset. The SMU connects to the port using the HW_DIR, ALTIN, HW_EN, HW_OUT signals. When the HW_EN port input is driven active by the SMU, SMU gets full control over the port data path, bypassing any other software configuration related to the usage of the ALTx inputs.

Figure 10-3 "SMU Data Path Interfaces with the Ports" on Page 10-10 fully specifies the data path connectivity of the SMU with the Ports. The mapping to the port for the product can be found in the port specification chapter of the TC21x/TC22x/TC23x microcontroller.

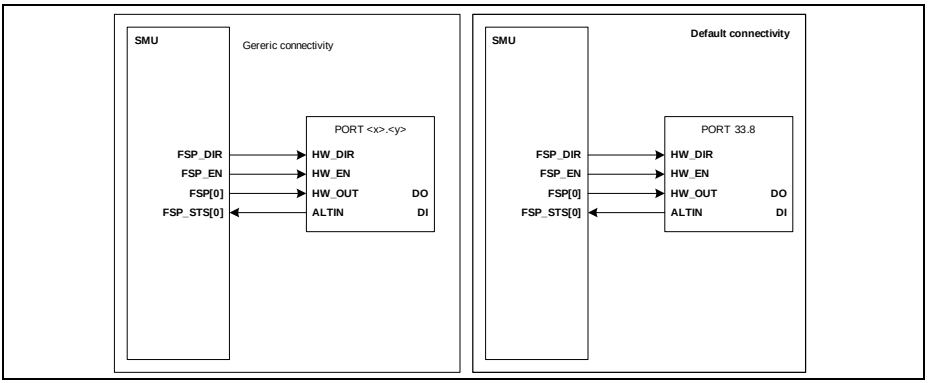


Figure 10-3 SMU Data Path Interfaces with the Ports

Figure 10-4 “SMU/PAD Control Interface to the PADs” on Page 10-10 provides a more detailed overview of the PORT structure and highlights the signals that play a role in the SMU connectivity.

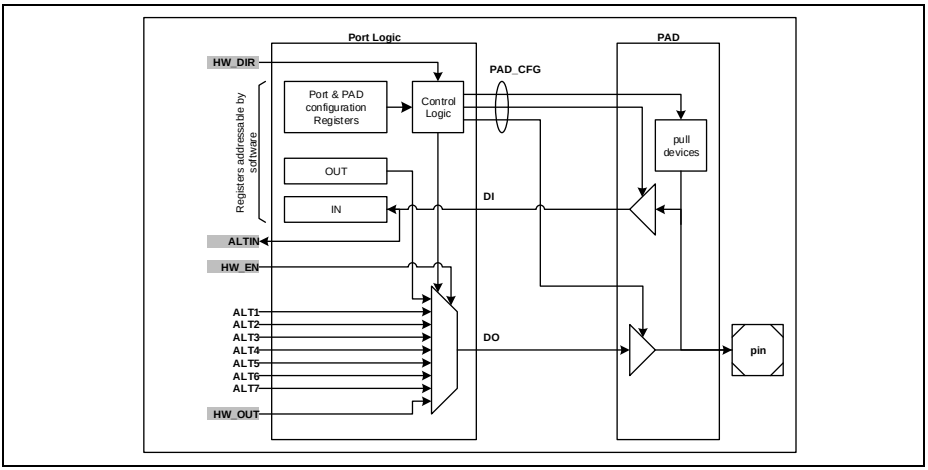


Figure 10-4 SMU/PAD Control Interface to the PADs

While FSP[0] is controlled by hardware, the FSP_DIR and FSP_EN are controlled by software as follows:

- FSP_DIR output is directly driven by the **SMU_PCTL** register HWDIR field
- FSP_EN output is directly driven by the **SMU_PCTL** register HWEN field

SMU_PCTL provides a field PCS that enables software to change the PAD control. With **SMU_PCTL** HWDIR, HWEN and PCS fields, software can control the PAD ownership transition from GPIO to full SMU hardware control following the guidelines provided by the **SMU_PCTL** register description.

To satisfy safety requirements it shall be ensured that the PAD operation is not affected by an application or system reset: the application can configure a system reset and the SMU Error Pin fault state activation upon detection of a critical alarm. This is achieved by safeguarding all the PAD configuration signals (see PAD_CFG signals **Figure 10-4**) into a special register. The contents of this register are exposed to software using the **SMU_PCTL**.PCFG field. They are only available for debug purposes.

The contents of the **SMU_PCTL** register are locked by the **SMU_KEYS** register and are only reset by a power-on reset, therefore the PAD configuration remains stable even in the presence of an application or system reset. Furthermore the **SMU_PCTL** register is implemented using dedicated Safety Flip-Flops safety mechanism that detects in run-time any bit change caused by a random hardware fault.

Refer to **“SMU Integration Guidelines” on Page 10-13** for the SMU and PORT configuration steps that are required to use the Error Pin.

10.4.2.4 Interface to the Safety Flip-Flop Safety Mechanism

The SMU provides a register interface that enables to test the Safety Flip-Flop safety mechanism. The register interface is realized by the **SMU_RMCTL**, **SMU_RMEF**, **SMU_RMSTS** registers.

10.4.3 SMU Integration Guidelines

This chapters extends the **"Interfaces Overview" on Page 10-8** section by providing additional information for the usage of the Error Pin (**"Fault Signaling Protocol (FSP)" on Page 10-52**) in combination with other input/output (GPIO) functions of the microcontroller.

Note: The PAD properties (push-pull, open-drain, drive strength,...) are configured in the registers of the PORT to which the SMU connects to. These registers are described in the PORT chapter of the microcontroller.

- During power-on-reset, the Error Pin is in high impedance: the pull devices are disabled under hardware control.
- After power-on-reset the default mode of the PORT to which the Error Pin is connected is GPIO. In that mode the pull devices can be used.
- Before changing the ownership of the PAD to SMU, software shall configure the PORT registers including the following:
  - Software shall disable the pull devices if GPIO is not used
  - Software shall program the GPIO registers of the error pin to strong driver output constant low
- To enable SMU to control the Error Pin PAD, software shall activate the PAD configuration safeguarding process (see **"Interface to the Ports (Error Pin)" on Page 10-8**).
  - The safeguarding process requires a software action that consists on writing a 1 into the **SMU_PCTL**.PCS field. **Only with the first transition from 0 to 1 leads to the safeguarding process. A new PORT configuration followed by a new transition from 0 to 1 of the SMU_PCTL.PCS has no effect to the hardware.**

10.4.4 Alarm Mapping

This section defines the mapping between the alarm signals at the input of the SMU and the alarm configuration registers. For that purpose alarm groups are defined. There is a one to one relationship between an alarm group index ALM<n>[index] signal and the alarm configuration and status registers (AG<n>[index]). A group is made of up to 32 input alarms; for convenience some entries may be reserved.

10.4.4.1 Pre-alarm Definition

There are situations where it is not necessary to implement configuration and status registers for every alarm; a typical case is a module with multiple SRAM instances. For that purpose a so-called PreAlarm[] bus is available to which the alarms that can be combined together are connected to. The result of the combination is sent to a HwAlarmOut[] output so that HwAlarmOut[] = Combine(PreAlarm[n], PreAlarm[m],...). The tables in **“Pre-alarm Signals” on Page 10-14** fully specify the arguments for the Combine() functions. The Combine() function implements a logical OR between all arguments.

Note: The unspecified (or unused) PreAlarm[] bits shall be tied to the logic level ‘0’ at the SMU input.

Note: The unspecified (or unused) HwAlarmOut[] bits shall be driven to the logic level ‘0’ by the SMU.

10.4.4.2 Pre-alarm Signals

In the next tables the PreAlarm[] bits are specified in the form index=MODULE.(type of error message), where MODULE.(type of error message) enables to identify the module instance and the nature of the error message. The error message is specified in a functional way and does not refer to a real signal name in the design. The modules connecting to the SMU shall use the same functional name to enable a cross-reference through the overall specification.

Table 10-2 HwAlarmOut[3:0]: CPU0 DCACHE/DSPR SRAM

Function
HwAlarmOut[0]= PreAlarm[0=CPU0.DMI.DSPR.(ECC single bit correction)]
HwAlarmOut[1]= PreAlarm[2=CPU0.DMI.DSPR.(ECC uncorrectable error)]

Table 10-2 HwAlarmOut[3:0]: CPU0 DCACHE/DSPR SRAM (cont'd)

Function
HwAlarmOut[2]= PreAlarm[4=CPU0.DMI.DSPR.(Address error)]
HwAlarmOut[3]= PreAlarm[6=CPU0.DMI.DSPR.(Address buffer overflow)]

Table 10-3 HwAlarmOut[7:4]: CPU1 DCACHE/DSPR SRAM

Function
HwAlarmOut[4]= PreAlarm[8=CPU1.DMI.DSPR.(ECC single bit correction)]
HwAlarmOut[5]= PreAlarm[10=CPU1.DMI.DSPR.(ECC uncorrectable error)]
HwAlarmOut[6]= PreAlarm[12=CPU1.DMI.DSPR.(Address error)]
HwAlarmOut[7]= PreAlarm[14=CPU1.DMI.DSPR.(Address buffer overflow)]

Table 10-4 HwAlarmOut[15:8]: CPU2 SRAMs

Function
HwAlarmOut[8]= PreAlarm[16=CPU2.DMI.DSPR.(ECC single bit correction)]
HwAlarmOut[9]= PreAlarm[18=CPU2.DMI.DSPR.(ECC uncorrectable error)]
HwAlarmOut[10]= PreAlarm[20=CPU2.DMI.DSPR.(Address error)]
HwAlarmOut[11]= PreAlarm[22=CPU2.DMI.DSPR.(Address buffer overflow)]

Table 10-5 HwAlarmOut[19:16]: GTM SRAMs

Function
HwAlarmOut[16]= PreAlarm[32=GTM.SRAM.FIFO.(ECC single bit correction)] or PreAlarm[33=GTM.SRAM.RAM0.(ECC single bit correction)] or PreAlarm[34=GTM.SRAM.RAM1.(ECC single bit correction)] or PreAlarm[35=GTM.SRAM.RAM1a.(ECC single bit correction)] or PreAlarm[36=GTM.SRAM.RAM1b.(ECC single bit correction)] or PreAlarm[37=GTM.SRAM.RAM2.(ECC single bit correction)]
HwAlarmOut[17=GTM.SRAM.(ECC uncorrectable error)]= PreAlarm[38=GTM.SRAM.FIFO.(ECC uncorrectable error)] or PreAlarm[39=GTM.SRAM.RAM0.(ECC uncorrectable error)] or PreAlarm[40=GTM.SRAM.RAM1.(ECC uncorrectable error)] or PreAlarm[41=GTM.SRAM.RAM1a.(ECC uncorrectable error)] or PreAlarm[42=GTM.SRAM.RAM1b.(ECC uncorrectable error)] or PreAlarm[43=GTM.SRAM.RAM2.(ECC uncorrectable error)]
HwAlarmOut[18]= PreAlarm[44=GTM.SRAM.FIFO.(Address error)] or PreAlarm[45=GTM.SRAM.RAM0.(Address error)] or PreAlarm[46=GTM.SRAM.RAM1.(Address error)] or PreAlarm[47=GTM.SRAM.RAM1a.(Address error)] or PreAlarm[48=GTM.SRAM.RAM1b.(Address error)] or PreAlarm[49=GTM.SRAM.RAM2.(Address error)]
HwAlarmOut[19]= PreAlarm[50=GTM.SRAM.FIFO.(Address Buffer overflow)] or PreAlarm[51=GTM.SRAM.RAM0.(Address Buffer overflow)] or PreAlarm[52=GTM.SRAM.RAM1.(Address Buffer overflow)] or PreAlarm[53=GTM.SRAM.RAM1a.(Address Buffer overflow)] or PreAlarm[54=GTM.SRAM.RAM1b.(Address Buffer overflow)] or PreAlarm[55=GTM.SRAM.RAM2.(Address Buffer overflow)]

Table 10-6 HwAlarmOut[23:20]: ERAY (FLEXRAY) SRAMs

Function
HwAlarmOut[20]= PreAlarm[56=ERAY.SRAM.OBF0.(ECC single bit correction)] or PreAlarm[58=ERAY.SRAM.TBF_IBF0.(ECC single bit correction)] or PreAlarm[60=ERAY.SRAM.MBF0.(ECC single bit correction)]
HwAlarmOut[21]= PreAlarm[62=ERAY.SRAM.OBF0.(ECC uncorrectable error)] or PreAlarm[64=ERAY.SRAM.TBF_IBF0.(ECC uncorrectable error)] or PreAlarm[66=ERAY.SRAM.MBF0.(ECC uncorrectable error)]
HwAlarmOut[22]= PreAlarm[68=ERAY.SRAM.OBF0.(Address error)] or PreAlarm[70=ERAY.SRAM.TBF_IBF0.(Address error)] or PreAlarm[72=ERAY.SRAM.MBF0.(Address error)]
HwAlarmOut[23]= PreAlarm[74=ERAY.SRAM.OBF0.(Address Buffer overflow)] or PreAlarm[76=ERAY.SRAM.TBF_IBF0.(Address Buffer overflow)] or PreAlarm[78=ERAY.SRAM.MBF0.(Address Buffer overflow)]

Table 10-7 HwAlarmOut[27:24]: CAN SRAM

Function
HwAlarmOut[24]= PreAlarm[80=CAN.SRAM.MCAN0.(ECC single bit correction)]
HwAlarmOut[25]= PreAlarm[82=CAN.SRAM.MCAN0.(ECC uncorrectable error)]
HwAlarmOut[26]= PreAlarm[84=CAN.SRAM.MCAN0.(Address error)]
HwAlarmOut[27]= PreAlarm[86=CAN.SRAM.MCAN0.(Address Buffer overflow)]

Table 10-8 HwAlarmOut[31:28]: LMU sub-system SRAMs

Function
HwAlarmOut[28]= PreAlarm[88=LMU.DAM.SRAM(ECC single bit correction)]
HwAlarmOut[29]= PreAlarm[90=LMU.DAM.SRAM(ECC uncorrectable error)]

Table 10-8 HwAlarmOut[31:28]: LMU sub-system SRAMs

Function
HwAlarmOut[30]= PreAlarm[92=LMU.DAM.SRAM(Address error)]
HwAlarmOut[31]= PreAlarm[94=LMU.DAM.SRAM(Address buffer overflow)]

Table 10-9 HwAlarmOut[34:32]: SRI Agents

Function
HwAlarmOut[32]= PreAlarm[101=PMU.SRI_SLAVE(SRI Address Phase Error)] PreAlarm[103=LMU.SRI_SLAVE(SRI Address Phase Error)] or PreAlarm[104=XBAR_SRI.SRI_SLAVE(SRI Address Phase Error)]
HwAlarmOut[33]= PreAlarm[110=PMU.SRI_SLAVE(SRI Write Data Phase Error)] or PreAlarm[112=LMU.SRI_SLAVE(SRI Write Data Phase Error)] or PreAlarm[113=XBAR_SRI.SRI_SLAVE(SRI Write Data Phase Error)]
HwAlarmOut[34]= PreAlarm[115=DMA.SRI_MASTER(SRI Read Data Phase Error)] or PreAlarm[116=HSSL.SRI_MASTER(SRI Read Data Phase Error)] or PreAlarm[117=SFI.SRI_MASTER(SRI Read Data Phase Error)] or PreAlarm[124=LMU.DAM.SRI_MASTER(SRI Read Data Phase Error)]
PreAlarm[109:105], PreAlarm[111], PreAlarm[114], PreAlarm[123:118] are reserved

Table 10-10 HwAlarmOut[35]: Safety Flip-Flop

Function
HwAlarmOut[35]= PreAlarm[125=SMU.(Safety FF Error Detection)] or PreAlarm[126=SCU.(Safety FF Error Detection)] or PreAlarm[127=MTU.(Safety FF Error Detection)]
Note: if a listed module does not implement a Safety FF, the corresponding input shall be tied to ‘0’.

Table 10-11 HwAlarmOut[44:41]: Misc. SRAMs

Function
HwAlarmOut[41]= PreAlarm[142=PMU.FSI.SRAM(ECC single bit correction)] or PreAlarm[143=DMA.SRAM(ECC single bit correction)] or PreAlarm[144=Ethernet.SRAM(ECC single bit correction)] or PreAlarm[145=PSI5.SRAM(ECC single bit correction)] or PreAlarm[146=MCDS.SRAM(ECC single bit correction)] or PreAlarm[147=CIF.SRAM(ECC single bit correction)] or PreAlarm[148=CIF.JPEG1_4.SRAM(ECC single bit correction)] or PreAlarm[149=CIF.JPEG3.SRAM(ECC single bit correction)]
HwAlarmOut[42]= PreAlarm[150=PMU.FSI.SRAM(ECC uncorrectable error)] or PreAlarm[151=DMA.SRAM(ECC uncorrectable error)] or PreAlarm[152=Ethernet.SRAM(ECC uncorrectable error)] or PreAlarm[153=PSI5.SRAM(ECC uncorrectable error)] or PreAlarm[154=MCDS.SRAM(ECC uncorrectable error)] or PreAlarm[155=CIF.SRAM(ECC uncorrectable error)] or PreAlarm[156=CIF.JPEG1_4.SRAM(ECC uncorrectable error)] or PreAlarm[157=CIF.JPEG3.SRAM(ECC uncorrectable error)]
HwAlarmOut[43]= PreAlarm[158=PMU.FSI.SRAM(Address error)] or PreAlarm[159=DMA.SRAM(Address error)] or PreAlarm[160=Ethernet.SRAM(Address error)] or PreAlarm[161=PSI5.SRAM(Address error)] or PreAlarm[162=MCDS.SRAM(Address error)] or PreAlarm[163=CIF.SRAM(Address error)] or PreAlarm[164=CIF.JPEG1_4.SRAM(Address error)] or PreAlarm[165=CIF.JPEG3.SRAM(Address error)]
HwAlarmOut[44]= PreAlarm[166=PMU.FSI.SRAM(Address Buffer overflow)] or PreAlarm[167=DMA.SRAM(Address Buffer overflow)] or PreAlarm[168=Ethernet.SRAM(Address Buffer overflow)] or PreAlarm[169=PSI5.SRAM(Address Buffer overflow)] or PreAlarm[170=MCDS.SRAM(Address Buffer overflow)] or PreAlarm[171=CIF.SRAM(Address Buffer overflow)] or PreAlarm[172=CIF.JPEG1_4.SRAM(Address Buffer overflow)] or PreAlarm[173=CIF.JPEG3.SRAM(Address Buffer overflow)]

Table 10-11 HwAlarmOut[44:41]: Misc. SRAMs (cont'd)

Function
HwAlarmOut[41]= PreAlarm[142=PMU.FSI.SRAM(ECC single bit correction)] or PreAlarm[143=DMA.SRAM(ECC single bit correction)] or PreAlarm[144=Ethernet.SRAM(ECC single bit correction)] or PreAlarm[146=MCDS.SRAM(ECC single bit correction)] or PreAlarm[147=CIF.SRAM(ECC single bit correction)] or PreAlarm[148=CIF.JPEG1_4.SRAM(ECC single bit correction)] or PreAlarm[149=CIF.JPEG3.SRAM(ECC single bit correction)]
Note: HwAlarmOut[44:41] connects respectively to ALM4[19:16]

Table 10-12 HwAlarmOut[45]: Watchdogs timeout

Function
HwAlarmOut[45]= PreAlarm[174=WDTCPU0.(timeout)] or PreAlarm[175=WDTCPU1.(timeout)] or PreAlarm[176=WDTCPU2.(timeout)] or PreAlarm[177=WDTS.(timeout)]

Table 10-13 HwAlarmOut[50:46]: PMU Alarms

Function
HwAlarmOut[46]= PreAlarm[178=PMU.PFLASH0.(ECC single bit correction notification)] or PreAlarm[179=PMU.PFLASH1.(ECC single bit correction notification)]
HwAlarmOut[47]= PreAlarm[186=PMU.PFLASH0.(ECC double bit correction notification)] or PreAlarm[187=PMU.PFLASH1.(ECC double bit correction notification)]
HwAlarmOut[48]= PreAlarm[194=PMU.PFLASH0.(ECC non correctable multiple bit error)] or PreAlarm[195=PMU.PFLASH1.(ECC non correctable multiple bit error)]
HwAlarmOut[49]= PreAlarm[202=PMU.PFLASH0.(Addressing error)] or PreAlarm[203=PMU.PFLASH1.(Addressing error)]
HwAlarmOut[50]= PreAlarm[210=PMU.PFLASH0.(Address buffer full)] or PreAlarm[211=PMU.PFLASH1.(Address buffer full)]

Table 10-14 HwAlarmOut[51]: Access Enable Alarms

Function
HwAlarmOut[51]= PreAlarm[218=IR.(Access Enable Error)][1] or PreAlarm[219=SCU.DTS.(Access Enable Error)]
PreAlarm[229:220] bits are reserved for extensions of the Access Enable Alarms

1) This alarm is related to the event where a master not configured in the module ACCEN register tries to access a register protected by the ACCEN scheme. Depending on the module all or only a subset of the module registers are protected by the ACCEN. Most of the modules only generate a bus error condition and no dedicated ACCEN alarm when the aforementioned event happens.

10.4.4.3 Non-compliant Alarms

This section describes the alarms that are not compliant with the SMU alarm protocol and for which a special processing is required. They are first connected to a pre-alarm to be processed by the SMU in order to deliver a SMU compatible input alarm signal.

Table 10-15 HwAlarmOut[63] specification: SCR Alarm

Function
HwAlarmOut[63]= PreAlarm[255=SCR(Standby Controller Alarm)]
PreAlarm[255] is a signal clocked by the fBACK clock, it shall be synchronized to the fSPB clock. HwAlarmOut[63] shall be connected to ALM3[24].

Note: The Standby Controller is not available in all devices. When not available PreAlarm[255] shall be connected to '0' and ALM3[24] shall be reserved.

10.4.4.4 Internal SMU Alarms

The SMU has also the capability to generate its own alarm signals. They are specified in **Table 10-16 “Internal Alarms” on Page 10-23**.

Table 10-16 Internal Alarms

HwAlarmOut	Function
HwAlarmOut[60]	Recovery Timer 0 Timeout
HwAlarmOut[61]	Recovery Timer 1 Timeout
HwAlarmOut[62]	FSP Fault State Activation
HwAlarmOut[63]	Reserved

10.4.4.5 Alarm Signals

The following tables fully specify the mapping between the alarms provided by the safety mechanisms implemented by the microcontroller and the alarm groups. Cells in **bold** refer to alarms coming from multiple instances of the same safety mechanism that are logically combined to a single alarm definition at the SMU level.

In the following tables the column “Safety Mechanism & Error Indication” indicates to which safety mechanism the alarm is related to. If multiple safety mechanisms are indicated, the alarm corresponds to the detection of an error by one of the listed safety mechanisms.

For some safety mechanisms different terms are used in the microcontroller documents; the following list provides a guideline between the term used in the alarm tables and the other definitions. In bold the definition used in the alarm tables.

- **Register Access Protection** or alternatively called Safety Register Protection
  - Purpose: Monitors the master identifier of a given bus-master during a write access to a configuration register. The master identifier is a hard-coded information that is provided during any bus access. If the master identifier is not enabled by the Register Access Protection configuration registers (ACCEN0) the write is aborted. Most of the modules do not provide a dedicated alarm for this event and instead will generate a bus error. Therefore the Register Access Protection is only documented where a dedicated alarm is available.
  - Note: for peripherals that implement memory-mapped SRAMs, the write accesses to the memories are monitored as well.
- **Bus-level Memory Protection Unit** (MPU) or alternatively called Safety Memory Protection
  - Purpose: Monitors the master identifier and the address of a given bus-master during a write access to a local SRAM. The master identifier is a hard-coded information that is provided during any bus access. If the master identifier is enabled by the Bus-level MPU configuration registers and the address is within the valid address range the write is accepted, otherwise the write is aborted and a Bus-level MPU alarm is issued.
  - The SRAMs monitored are the {PSPR, DSPR} SRAMs of each CPU and the LMU SRAMs when available in the product.

Table 10-17 TC21x/TC22x/TC23x **Alarm Mapping related to Alarm 0 group**

Alarm Index	Module	Safety Mechanism & Alarm Indication
ALM0[0]	CPU0	Safety Mechanism: Lockstep CPU Alarm: Lockstep Comparator Error
ALM0[1]	CPU0	Safety Mechanism: Bus-level Memory Protection Unit Alarm: Bus-level MPU violation. Safety Mechanism: Register Access Protection Alarm: Access Protection violation.
ALM0[2]	Reserved	Reserved
ALM0[3]	CPU0	Safety Mechanism: SRAM Error Detection Code (EDC) Alarm: PCACHE TAG uncorrectable error
ALM0[4]	CPU0	Safety Mechanism: SRAM Address Monitor Alarm: PCACHE TAG address error
ALM0[5]	CPU0	Safety Mechanism: SRAM Monitor Alarm: PCACHE TAG Address Buffer overflow
ALM0[6]	CPU0	Safety Mechanism: SRAM ECC Alarm: Unified PCACHE/PSPR single-bit correction
ALM0[7]	CPU0	Safety Mechanism: SRAM ECC Alarm: Unified PCACHE/PSPR uncorrectable error
ALM0[8]	CPU0	Safety Mechanism: SRAM Address Monitor Alarm: Unified PCACHE/PSPR Address error
ALM0[9]	CPU0	Safety Mechanism: SRAM Monitor Alarm Unified PCACHE/PSPR Address Buffer overflow
ALM0[10]	**CPU0**	**Safety Mechanism(s): SRAM ECC Alarm: Unified DCACHE/DSPR single-bit correction**
ALM0[11]	**CPU0**	**Safety Mechanism(s): SRAM ECC Alarm: Unified DCACHE/DSPR uncorrectable error**
ALM0[12]	**CPU0**	**Safety Mechanism(s): SRAM Address Monitor Alarm: Unified DCACHE/DSPR address error**
ALM0[13]	**CPU0**	**Safety Mechanism(s): SRAM Monitor Alarm: Unified DCACHE/DSPR Address Buffer overflow**
ALM[17:14]	Reserved	Reserved
ALM0[18]	CPU0	Safety Mechanism: SRI End-to-End EDC Alarm: CPU Instruction Fetch SRI Interface EDC Error

Table 10-17 TC21x/TC22x/TC23x **Alarm Mapping related to Alarm 0 group** (cont'd)

Alarm Index	Module	Safety Mechanism & Alarm Indication
ALM0[19]	CPU0	Safety Mechanism: SRI End-to-End EDC Alarm: CPU Data SRI Interface (Load/Store) EDC Error
ALM0[31:20]	Reserved	Reserved

Table 10-18 TC21x/TC22x/TC23x **Alarm Mapping for related to ALM1 group**

Alarm Index	Module	Description
ALM1[0]	CPU1	Safety Mechanism: Lockstep CPU Alarm: Lockstep Comparator Error
ALM1[1]	CPU1	Safety Mechanism: Bus-level Memory Protection Unit Alarm: Bus-level MPU violation. Safety Mechanism: Register Access Protection Alarm: Access Protection violation.
ALM1[2]	Reserved	Reserved
ALM1[3]	CPU1	Safety Mechanism: SRAM Error Detection Code (EDC) Alarm: PCACHE TAG uncorrectable error
ALM1[4]	CPU1	Safety Mechanism: SRAM Address Monitor Alarm: PCACHE TAG SRAM address error
ALM1[5]	CPU1	Safety Mechanism: SRAM Monitor Alarm: PCACHE TAG SRAM address buffer overflow
ALM1[6]	CPU1	Safety Mechanism: SRAM Error Correction Code (ECC) Alarm: Unified PCACHE/PSPR single-bit correction
ALM1[7]	CPU1	Safety Mechanism: SRAM Error Correction Code (ECC) Alarm: Unified PCACHE/PSPR uncorrectable error
ALM1[8]	CPU1	Safety Mechanism: SRAM Address Monitor Alarm: Unified PCACHE/PSPR address error
ALM1[9]	CPU1	Safety Mechanism: SRAM Monitor Alarm: Unified PCACHE/PSPR address buffer overflow
ALM1[13:10] connects to pre-alarms specified by Table 10-3 “HwAlarmOut[7:4]: CPU1 DCACHE/DSPR SRAM” on Page 10-15		
ALM1[10]	**CPU1**	Safety Mechanism: SRAM Error Correction Code (ECC) Alarm: Unified DCACHE/DSPR single-bit correction
ALM1[11]	**CPU1**	Safety Mechanism: SRAM Error Correction Code (ECC) Alarm: Unified DCACHE/DSPR uncorrectable error
ALM1[12]	**CPU1**	Safety Mechanism: SRAM Address Monitor Alarm: Unified DCACHE/DSPR address error
ALM1[13]	**CPU1**	Safety Mechanism: SRAM Monitor Alarm: Unified DCACHE/DSPR address buffer overflow
ALM1[14]	CPU1	Safety Mechanism: SRAM Error Correction Code (ECC) Alarm: DCACHE TAG SRAM correction

Table 10-18 TC21x/TC22x/TC23x **Alarm Mapping for related to ALM1 group**

Alarm Index	Module	Description
ALM1[15]	CPU1	Safety Mechanism: SRAM Error Correction Code (ECC) Alarm: DCACHE TAG SRAM uncorrectable error
ALM1[16]	CPU1	Safety Mechanism: SRAM Address Monitor Alarm: DCACHE TAG SRAM address error
ALM1[17]	CPU1	Safety Mechanism: SRAM Monitor Alarm: DCACHE TAG SRAM address buffer overflow
ALM1[18]	CPU1	Safety Mechanism: SRI End-to-End EDC Alarm: CPU Instruction Fetch SRI Interface EDC Error
ALM1[19]	CPU1	Safety Mechanism: SRI End-to-End EDC Alarm: CPU Data SRI Interface (Load/Store) EDC Error
ALM1[31:20]	Reserved	Reserved

Table 10-19 TC21x/TC22x/TC23x **Alarm Mapping related to ALM2 group**

Alarm Index	Module	Description
ALM2[0]	Reserved	Reserved
ALM2[1]	Reserved	Reserved
ALM2[6:2] connects to pre-alarms specified by **Table 10-13 "HwAlarmOut[50:46]: PMU Alarms" on Page 10-20**		
ALM2[2]	**PMU**	**Safety Mechanism: PFLASH ECC** **Alarm: single bit correction**
ALM2[3]	**PMU**	**Safety Mechanism: PFLASH ECC** **Alarm: double bit correction**
ALM2[4]	**PMU**	**Safety Mechanism: PFLASH ECC** **Alarm: non correctable multiple bit**
ALM2[5]	**PMU**	**Safety Mechanism: PFLASH** **Alarm: Addressing error**
ALM2[6]	**PMU**	**Safety Mechanism: PFLASH Monitor** **Alarm: address buffer full**
ALM2[7]	PMU	Safety Mechanism: PFLASH ECC monitor Alarm: PFLASH ECC error Note: All the PFLASH ECC monitor alarms are combined into a single alarm to the SMU.
ALM2[8]	PMU	Safety Mechanism: PFLASH EDC monitor Alarm: EDC comparator error Note: All the PFLASH EDC monitor alarms are combined into a single alarm to the SMU.
ALM2[9]	Reserved	Reserved
ALM2[10]	Reserved	Reserved
ALM2[11]	Reserved	Reserved
ALM2[12]	Reserved	Reserved
ALM2[13]	Reserved	Reserved
ALM2[14]	Reserved	Reserved
ALM2[15]	LMU	Safety Mechanism: SRAM ECC Monitor Alarm: ECC Error
ALM2[16]	LMU	Safety Mechanism: Register Access Protection Alarm: Register Access Protection error Safety Mechanism: Bus-level MPU Alarm: Bus-level MPU error

Table 10-19 TC21x/TC22x/TC23x **Alarm Mapping related to ALM2 group** (cont'd)

Alarm Index	Module	Description
ALM2[20:17] connects to pre-alarms specified by **Table 10-8 "HwAlarmOut[31:28]: LMU sub-system SRAMs" on Page 10-17**		
ALM2[17]	**LMU**	**Safety Mechanism: SRAM Error Correction Code (ECC)** **Alarm: SRAM single-bit correction**
ALM2[18]	**LMU**	**Safety Mechanism: SRAM Error Correction Code (ECC)** **Alarm: SRAM uncorrectable error**
ALM2[19]	**LMU**	**Safety Mechanism: SRAM Address Monitor** **Alarm: SRAM Address error**
ALM2[20]	**LMU**	**Safety Mechanism: SRAM Monitor** **Alarm: SRAM Address buffer overflow**
ALM2[23:21] connects to pre-alarms specified by Table 10-9 "HwAlarmOut[34:32]: SRI Agents" on Page 10-18		
ALM2[21]	**SRI**	**Safety Mechanism: SRI Error Detection Code (EDC)** **Alarm: EDC Address phase error** **Scope: non-CPU modules connected to SRI**
ALM2[22]	**SRI**	**Safety Mechanism: SRI Error Detection Code (EDC)** **Alarm: EDC Write phase error** **Scope: non-CPU modules connected to SRI**
ALM2[23]	**SRI**	**Safety Mechanism: SRI Error Detection Code (EDC)** **Alarm: EDC Read phase error** **Scope: non-CPU modules connected to SRI**
ALM2[24]	Reserved	Reserved
ALM2[25]	IR	Safety Mechanism: IR Monitor (Error Detection Code covering Configuration & Data Path) Alarm: EDC error
ALM2[26]	IOM	Safety Mechanism: Input/Output Monitor (IOM) Alarm: Pin Mismatch Indication
ALM2[27]	Reserved	Reserved
ALM2[28]	Reserved	Reserved
ALM2[29]	SMU	Safety Mechanism: Recovery Timer 0 Alarm: Timer time-out

Table 10-19 TC21x/TC22x/TC23x **Alarm Mapping related to ALM2 group** (cont'd)

Alarm Index	Module	Description
ALM2[30]	SMU	Safety Mechanism: Recovery Timer 1 Alarm: Timer time-out
ALM2[31]	SMU	Design Measure: ErrorPin Alarm: ErrorPin Fault State Activation This feature enables software to get a notification when the Error Pin is activated by hardware.

Table 10-20 TC21x/TC22x/TC23x **Alarm Mapping related to ALM3 group**

Alarm Index	Module	Description
ALM3[0]	SCU/CGU	Design Measure: System PLL Oscillator Watchdog (OSC_WDT) Alarm: input clock out of range
ALM3[1]	SCU/CGU	Design Measure: PLL loss-of-lock detection Alarm: System PLL VCO Loss-of-Lock Event
ALM3[2]	SCU/CGU	Design Measure: PLL loss-of-lock detection Alarm: PLL_ERAY VCO Loss-of-Lock Event
ALM3[3]	SCU/CGU	Safety Mechanism: Clock Monitor Alarm: STM clock out of range frequency
ALM3[4]	SCU/CGU	Safety Mechanism: Clock Monitor Alarm: PLL_ERAY out of range frequency
ALM3[5]	SCU/CGU	Safety Mechanism: Clock Monitor Alarm: System PLL out of range frequency
ALM3[6]	SCU/CGU	Safety Mechanism: Clock Monitor Alarm: SRI clock out of range frequency
ALM3[7]	SCU/CGU	Safety Mechanism: Clock Monitor Alarm: SPB clock out of range frequency
ALM3[8]	SCU/CGU	Safety Mechanism: Clock Monitor Alarm: GTM clock out of range frequency
ALM3[9]	SCU/CGU	Safety Mechanism: Clock Monitor Alarm: ADC clock out of range frequency
ALM3[10]	Reserved	Reserved
ALM3[11]	SCU/EVR	Safety Mechanism: Voltage Monitor Alarm: EVR 1.3V under voltage
ALM3[12]	SCU/EVR	Safety Mechanism: Voltage Monitor Alarm: EVR 1.3V over voltage
ALM3[13]	SCU/EVR	Safety Mechanism: Voltage Monitor Alarm: EVR 3.3V under voltage
ALM3[14]	SCU/EVR	Safety Mechanism: Voltage Monitor Alarm: EVR 3.3V over voltage
ALM3[15]	SCU/EVR	Safety Mechanism: Voltage Monitor Alarm: External Supply under voltage

Table 10-20 TC21x/TC22x/TC23x **Alarm Mapping related to ALM3 group** (cont'd)

Alarm Index	Module	Description
ALM3[16]	SCU/EVR	Safety Mechanism: Voltage Monitor Alarm: External Supply over voltage
ALM3[17]	SCU/WDTS	Safety Mechanism: Safety Watchdog Alarm: watchdog time-out
ALM3[18]	SCU/WDTCPU0	Safety Mechanism: CPU0 Watchdog Alarm: watchdog time-out
ALM3[19]	SCU/WDTCPU1	Safety Mechanism: CPU1 Watchdog Alarm: Watchdog time-out
ALM3[20]	SCU/WDTCPU2	Safety Mechanism: CPU2 Watchdog Alarm: watchdog time-out
ALM3[21] connects to pre-alarms specified by Table 10-12 "HwAlarmOut[45]: Watchdogs timeout" on Page 10-20		
ALM3[21]	**SCU/WDT**	**Safety Mechanism: All Watchdogs Alarm: watchdog time-out. This alarm is a logical OR over all watchdog time-out alarms.**
ALM3[22] connects to pre-alarms specified by Table 10-14 "HwAlarmOut[51]: Access Enable Alarms" on Page 10-21		
ALM3[22]	**SPB Peripherals**	**Safety Mechanism: Register Access Protection** Alarm: Access Enable Error
ALM3[23]	Reserved	Reserved
ALM3[24]	Reserved	Reserved
ALM3[25]	SCU/DTS	Design Measure: Die Temperature Sensor Alarm: Temperature underflow
ALM3[26]	SCU/DTS	Design Measure: Die Temperature Sensor Alarm: Temperature overflow
ALM3[27] connects to pre-alarms specified by Table 10-10 "HwAlarmOut[35]: Safety Flip-Flop" on Page 10-18		
ALM3[27]	**Registers**	**Safety Mechanism: Safety Flip-Flop (redundant or triple-modular register structure). Alarm: Safety Flip-Flop error detection**
ALM3[28]	Reserved	Reserved
ALM3[29]	SCU	Design Measure: Emergency Stop Alarm: External Emergency Stop Signal Event

Table 10-20 TC21x/TC22x/TC23x **Alarm Mapping related to ALM3 group** (cont'd)

Alarm Index	Module	Description
ALM3[30]	SRI	Design Measure: Built-in SRI Error Detection Alarm: SRI Bus error event
ALM3[31]	SPB	Design Measure: Built-in SPB Error Detection Alarm: SPB Bus error The SPB bus error can result from multiple root causes: protocol violation, incorrect address, register access protection violation

Table 10-21 TC21x/TC22x/TC23x **Alarm Mapping related to ALM4 group**

Alarm Index	Module	Description
ALM4[3:0] connects to pre-alarms specified by **Table 10-5 "HwAlarmOut[19:16]: GTM SRAMs" on Page 10-16**		
ALM4[0]	**GTM**	**Safety Mechanism: SRAM Error Correction Code (ECC) Alarm: SRAM single-bit correction**
ALM4[1]	**GTM**	**Safety Mechanism: SRAM Error Correction Code (ECC) Alarm: SRAM uncorrectable error**
ALM4[2]	**GTM**	**Safety Mechanism: SRAM Address Monitor Alarm: SRAM address error**
ALM4[3]	**GTM**	**Safety Mechanism: SRAM Monitor SRAM Address Buffer overflow**
ALM4[7:3] connects to pre-alarms specified by **Table 10-7 "HwAlarmOut[27:24]: CAN SRAM" on Page 10-17**		
ALM4[4]	**CAN**	**Safety Mechanism: SRAM Error Correction Code (ECC) Alarm: SRAM single-bit correction**
ALM4[5]	**CAN**	**Safety Mechanism: SRAM Error Correction Code (ECC) Alarm: SRAM uncorrectable error**
ALM4[6]	**CAN**	**Safety Mechanism: SRAM Address Monitor Alarm: SRAM address error**
ALM4[7]	**CAN**	**Safety Mechanism: SRAM Monitor SRAM Address Buffer overflow**
ALM4[11:8] connects to pre-alarms specified by **Table 10-6 "HwAlarmOut[23:20]: ERAY (FLEXRAY) SRAMs" on Page 10-17**		
ALM4[8]	**FLEXRAY**	**Safety Mechanism: SRAM Error Correction Code (ECC) Alarm: SRAM single-bit correction**
ALM4[9]	**FLEXRAY**	**Safety Mechanism: SRAM Error Correction Code (ECC) Alarm: SRAM uncorrectable error**
ALM4[10]	**FLEXRAY**	**Safety Mechanism: SRAM Address Monitor Alarm: SRAM address error**
ALM4[11]	**FLEXRAY**	**Safety Mechanism: SRAM Monitor SRAM Address Buffer overflow**
ALM4[12]	EMEM	Safety Mechanism: SRAM Error Correction Code (ECC) Alarm: SRAM single-bit correction

Table 10-21 TC21x/TC22x/TC23x **Alarm Mapping related to ALM4 group** (cont'd)

Alarm Index	Module	Description
ALM4[13]	EMEM	Safety Mechanism: SRAM Error Correction Code (ECC) Alarm: SRAM ECC uncorrectable error
ALM4[14]	Reserved	Reserved
ALM4[15]	EMEM	Safety Mechanism: SRAM Monitor Alarm: SRAM Address Buffer overflow
ALM4[19:16] connects to pre-alarms specified by **Table 10-11 "HwAlarmOut[44:41]: Misc. SRAMs" on Page 10-19**		
ALM4[16]	**Misc SRAMs**[1)]	**Safety Mechanism: SRAM Error Correction Code (ECC)** **Alarm: SRAM single-bit correction**
ALM4[17]	**Misc.** SRAMs	**Safety Mechanism: SRAM Error Correction Code (ECC)** **Alarm: SRAM uncorrectable error**
ALM4[18]	**Misc.** SRAMs	**Safety Mechanism: SRAM Address Monitor** **Alarm: SRAM address error**
ALM4[19]	**Misc.** SRAMs	**Safety Mechanism: SRAM Address Monitor** **Alarm: SRAM Address Buffer overflow**
ALM4[31...20]	Reserved	Reserved

1) The list of miscellaneous SRAMs can be found in **Table 10-11 "HwAlarmOut[44:41]: Misc. SRAMs" on Page 10-19**.

A dedicated alarm group **Table 10-22 “TC21x/TC22x/TC23x Alarm Mapping related to ALM5 group” on Page 10-37** enables software to submit alarms and use the SMU infrastructure for alarm management. Alarms can be submitted using the register command interface **SMU_CMD**.

Table 10-22 TC21x/TC22x/TC23x Alarm Mapping related to ALM5 group

Alarm Index	Module	Description
ALM5[0]	Software	Safety Mechanism: Software Monitor Alarm: Software Alarm 0
ALM5[1]	Software	Safety Mechanism: Software Monitor Alarm: Software Alarm 1
ALM5[2]	Software	Safety Mechanism: Software Monitor Alarm: Software Alarm 2
ALM5[3]	Software	Safety Mechanism: Software Monitor Alarm: Software Alarm 3
ALM5[4]	Software	Safety Mechanism: Software Monitor Alarm: Software Alarm 4
ALM5[5]	Software	Safety Mechanism: Software Monitor Alarm: Software Alarm 5
ALM5[6]	Software	Safety Mechanism: Software Monitor Alarm: Software Alarm 6
ALM5[7]	Software	Safety Mechanism: Software Monitor Alarm: Software Alarm 7
ALM5[8]	Software	Safety Mechanism: Software Monitor Alarm: Software Alarm 8
ALM5[9]	Software	Safety Mechanism: Software Monitor Alarm: Software Alarm 9
ALM5[10]	Software	Safety Mechanism: Software Monitor Alarm: Software Alarm 10
ALM5[11]	Software	Safety Mechanism: Software Monitor Alarm: Software Alarm 11
ALM5[12]	Software	Safety Mechanism: Software Monitor Alarm: Software Alarm 12
ALM5[13]	Software	Safety Mechanism: Software Monitor Alarm: Software Alarm 13
ALM5[14]	Software	Safety Mechanism: Software Monitor Alarm: Software Alarm 14

Table 10-22 TC21x/TC22x/TC23x Alarm Mapping related to ALM5 group (cont'd)

Alarm Index	Module	Description
ALM5[15]	Software	Safety Mechanism: Software Monitor Alarm: Software Alarm 15
ALM5[31..16]	Reserved	Reserved

Table 10-23 TC21x/TC22x/TC23x **Alarm Mapping related to ALM6 group**

Date	Module	Description
ALM6[0]	Reserved	Reserved
ALM6[1]	CPU2	Safety Mechanism: Bus-level Memory Protection Unit Alarm: Bus-level MPU violation. Safety Mechanism: Register Access Protection Alarm: Access Protection violation.
ALM6[2]	Reserved	Reserved
ALM6[3]	CPU2	Safety Mechanism: SRAM Error Detection Code (EDC) Alarm: PCACHE TAG SRAM uncorrectable error
ALM6[4]	CPU2	Safety Mechanism: SRAM Address Monitor Alarm: PCACHE TAG SRAM address error
ALM6[5]	CPU2	Safety Mechanism: SRAM Monitor Alarm: PCACHE TAG SRAM address buffer overflow
ALM6[6]	CPU2	Safety Mechanism: SRAM Error Correction Code (ECC) Alarm: Unified PCACHE/PSPR single-bit correction
ALM6[7]	CPU2	Safety Mechanism: SRAM Error Correction Code (ECC) Alarm: Unified PCACHE/PSPR uncorrectable error
ALM6[8]	CPU2	Safety Mechanism: SRAM Address Monitor Alarm: Unified PCACHE/PSPR address error
ALM6[9]	CPU2	Safety Mechanism: SRAM Monitor Alarm: Unified PCACHE/PSPR address buffer overflow
ALM6[10]	CPU2	Safety Mechanism: SRAM Error Correction Code (ECC) Alarm: Unified DCACHE/DSPR single-bit correction
ALM6[11]	CPU2	Safety Mechanism: SRAM Error Correction Code (ECC) Alarm: Unified DCACHE/DSPR uncorrectable error
ALM6[12]	CPU2	Safety Mechanism: SRAM Address Monitor Alarm: Unified DCACHE/DSPR address error
ALM6[13]	CPU2	Safety Mechanism: SRAM Monitor Alarm: Unified DCACHE/DSPR address buffer overflow
ALM6[14]	CPU2	Safety Mechanism: SRAM Error Correction Code (ECC) Alarm: DCACHE TAG SRAM single-bit correction
ALM6[15]	CPU2	Safety Mechanism: SRAM Error Correction Code (ECC) Alarm: DCACHE TAG SRAM uncorrectable error
ALM6[16]	CPU2	Safety Mechanism: SRAM Address Monitor Alarm: DCACHE TAG SRAM Address error

Table 10-23 TC21x/TC22x/TC23x **Alarm Mapping related to ALM6 group** (cont'd)

Date	Module	Description
ALM6[17]	CPU2	Safety Mechanism: SRAM Monitor Alarm: DCACHE TAG SRAM Address buffer overflow
ALM6[18]	CPU2	Safety Mechanism: SRI End-to-End EDC Alarm: CPU Instruction Fetch SRI Interface EDC Error
ALM6[19]	CPU2	Safety Mechanism: SRI End-to-End EDC Alarm: CPU Data SRI Interface (Load/Store) EDC Error
ALM6[31..20]	Reserved	Reserved

10.4.5 Alarm Handling

This section specifies the hardware and software alarm processes.

10.4.5.1 Alarm protocol

Each safety mechanism shall interface with the SMU using a pre-defined protocol. The protocol enables to cross clock domains in a reliable manner. The operation of the protocol has no influence to the software layers.

10.4.5.2 Alarm Configuration

Upon reception of an alarm event the SMU decodes the actions to be performed. The action can be classified into an internal behavior and an external behavior. Both the internal and external behavior can be configured for every alarm.

The external behavior is related to the Fault Signaling Protocol (see **“Fault Signaling Protocol (FSP)” on Page 10-52**). The external behavior is configured via the following registers:

- **SMU_AG0FSP**
- **SMU_AG1FSP**
- **SMU_AG2FSP**
- **SMU_AG3FSP**
- **SMU_AG4FSP**
- **SMU_AG5FSP**
- **SMU_AG6FSP**

Alarm Action Configuration Registers

The internal behavior of the SMU under the presence of an alarm is controlled via the following registers:

- **SMU_AG0CFx (x=0-2)**
- **SMU_AG1CFx (x=0-2)**
- **SMU_AG2CFx (x=0-0)**, **SMU_AG2CFx (x=1-1)**, **SMU_AG2CFx (x=2-2)**
- **SMU_AG3CFx (x=0-0)**, **SMU_AG3CFx (x=1-1)**, **SMU_AG3CFx (x=2-2)**
- **SMU_AG4CFx (x=0-2)**
- **SMU_AG5CFx (x=0-2)**
- **SMU_AG6CFx (x=0-2)**

Alarm Action Configuration Codes

The internal behavior is specified by a 3-bit code as follows:

- Code = SMU_AG<n>CF2. SMU_AG<n>CF1. SMU_AG<n>CF0, n=0...6

Table 10-24 SMU Alarm Configuration

Code	Name	Behavior
0x0	SMU_NA	No Action. Reset value. Alarm disabled.
0x1	SMU_RSVD	Reserved. No Action. Alarm disabled.
0x2	SMU_IGCS0	Sends an interrupt request to the interrupt system according to the Interrupt Generation Configuration Set 0 from the **SMU_AGC** register.
0x3	SMU_IGCS1	Sends an interrupt request to the interrupt system according to the Interrupt Generation Configuration Set 1 from the **SMU_AGC** register.
0x4	SMU_IGCS2	Sends an interrupt request to the interrupt system according to the Interrupt Generation Configuration Set 2 from the **SMU_AGC** register.
0x5	SMU_NMI	Sends an NMI request to the SCU
0x6	SMU_RESET	Sends a reset request to the SCU. The SCU shall be configured to generate an application or system reset.
0x7	SMU_IDLE	Triggers a CPU idle request using Idle Configuration Set from the **SMU_AGC** register

Unless otherwise specified the reset state of the alarms is “No Action”. Refer to reset values of registers listed in **“Alarm Action Configuration Registers” on Page 10-41** and special cases described in **“Watchdog Alarms” on Page 10-46**.

10.4.5.3 Alarm operation

Whenever an input alarm event is detected and the SMU state machine is in the RUN or FAULT state, the SMU checks for the corresponding actions to be done for the internal action and for the FSP in a concurrent way. If an input alarm event is detected and no action is specified for the alarm, the corresponding bit shall be set to 1 as well but no action takes place.

The processing of the incoming alarm events is performed as follows:

- The alarm groups are scanned in a linear order starting with alarm group 0. The switching from one alarm group to the next one, takes several clock cycles.
- During the switch operation to a new alarm group (AGy) a snapshot of the pending alarms for AGy is made by hardware.
- If there are no pending alarms for the active alarm group AGy nothing happens during the processing time slot related to this alarm group.
- If there are pending alarms they are processed in a fixed order 0 up to 31.
  - If new alarms arrived for the active alarm group, they are only processed in the next time slot for the alarm group.
  - The processing of an alarm within an alarm group may take several fSPB cycles.
  - If the status flag related to a pending alarm event is already set to 1 the alarm event is ignored and the next available pending alarm within the active group is searched.
  - Whenever a pending alarm event is processed, the corresponding status bit is set to 1 by hardware in the AG<x> register. If an internal SMU action is configured the action counter (ACNT) in the **SMU_AFCNT** register is incremented.

10.4.5.4 Alarm Status Registers

Table 10-25 “Handling of Alarm Status” on Page 10-44 specifies the possible software actions on the AG<x> alarm group status registers depending on the SMU State machine state.

Table 10-25 Handling of Alarm Status

SMU State Machine	SW Action	Effect on AG<x>
START	SMU_ASCE(0) command Write **Data** at AG<x> Address	If (**Data[i]** == 1) AG<x>[i] = 0 else AG<x>[i] unchanged
START	Write **Data** at AG<x> Address	If (**Data[i]** == 1) AG<x>[i] = 1 else AG<x>[i] unchanged
RUN	SMU_ASCE(0) command Write **Data** at AG<x> Address	If (**Data[i]** == 1) AG<x>[i] = 0 else AG<x>[i] unchanged
RUN	Write **Data** at AG<x> Address	No Effect
FAULT	SMU_ASCE(0) command Write **Data** at AG<x> Address	If (**Data[i]** == 1) AG<x>[i] = 0 else AG<x>[i] unchanged
FAULT	Write **Data** at AG<x> Address	No Effect

In the START state software has the possibility to “emulate” the occurrence of input alarm events by writing at an AG<x> address. The resulting set of an AG<x> bit per software can take several hundreds of clock cycles depending on the hardware decoding state and the number of active alarms. Software shall read back the AG<x> register to ensure the completion of the operation if necessary.

10.4.5.5 Alarm Debug Registers

The Alarm debug registers enable the application to improve the diagnosis of the root cause that lead to a malfunction. In that context they may help to implement recovery strategies, if allowed by the application. The **SMU_ADx (x=0-6)** debug registers shall make a snapshot of the **SMU_AGx (x=0-6)** when:

- the action to be executed by the SMU is a reset when the SMU is in the RUN or in the FAULT state
- a SMU state machine (SSM) transition to the FAULT state takes place, either controlled by the SMU hardware or a software command

The **SMU_ADx (x=0-6)** shall only cleared by a power-on reset.

10.4.5.6 Port Emergency Stop

The port emergency stop feature enables forcing a pad into General Purpose Input Mode. The port emergency stop request to the SCU can be activated by any of the following situations:

- a SMU_ActivatePES() software command
- an alarm event with SMU_AG<x>FSP enabled and **SMU_FSP**.PES enabled
- an alarm event with an internal action configured in SMU_AG<x>CFx registers and SMU_AGC.PES enabled for that action.

10.4.5.7 Recovery Timer

A recovery timer (RT) is available to enable the monitoring of the duration or internal error handlers activated by an alarm NMI or Interrupt action. In the current SMU implementation two independent instances (RT0 and RT1) are available. The recovery timer duration (identical for all instances) is configured in the register **SMU_RTC**. It is possible to enable or disable each instance, however RT0 is enabled by default as it is required for the operation of the CPU watchdogs (see also **“Watchdog Alarms” on Page 10-46**). In addition to **SMU_RTC** an additional configuration register (SMU_RTC<n>) is available per instance to configure the alarm mapping.

The alarm mapping consists of a pair of parameters {GID<n>[2:0], ALID<n>[4:0]}, where GID<n>[2:0] is a group identifier and ALID<n>[4:0] is the alarm identifier belonging to the group. Four {GID[], ALID[]} pairs can be configured per recovery timer instance. It is possible to configure multiple time the same group identifier. If less than four alarms need to be mapped to a recovery timer, the same {GID[], ALID[]} shall be configured multiple times.

Note: The use of the recovery timer only makes sense if the internal action is an interrupt or NMI. However no hardware check is done, it is up to software to configure the SMU in the appropriate way.

If a recovery timer is enabled and for any of the {GID[], ALID[]} pairs and an alarm event occurs and if an internal action is configured resulting to an internal action (the alarm status shall be cleared) the recovery timer is automatically started by hardware. Such situation is called a recovery timer event. An alarm without internal action shall not start a recovery timer.

Once an recovery timer event is occurs, the recovery timer starts and counts until software stops it with the SMU_RTStop(). If the timer expires an internal SMU alarm (Recovery Timer Timeout) is issued. During the time the recovery timer is running any other action that requests the recovery timer is ignored. If such event happens, the bit RTME (Recovery Timer Missed Event) is set to ‘1’ by hardware in the **SMU_STS** register. The bit RTME can only be cleared by software. The bit RTS (Recovery Timer Status) is set to ‘1’ by hardware in the **SMU_STS** register during the time the recovery

timer is running: from the timer activation until a SMU_RTStop() is received or the timer expires. The bit RTS is cleared by hardware upon reception of SMU_RTStop() or the timer expires.

If a SMU_RTStop() command is received when the recovery timer is not active, the command returns an error response.

10.4.5.8 Watchdog Alarms

The watchdogs (WDT) timeout alarms require a special processing in order to ensure a correct microcontroller behavior if the watchdogs are not serviced by software or firmware. It shall be ensured that the microcontroller is reset after a pre-warning phase, where software can still perform some critical actions.

- Every Timeout Alarm shall activate an NMI
- Recovery Timer 0 shall be configured to service WDT timeout alarms
- Recovery Timer 0 timeout alarm shall be configured to issue a reset request and activate the Fault Signalling Protocol.

The aforementioned properties are implemented as reset values for the watchdog(s) timeout alarm(s) and for the recovery timer 0.

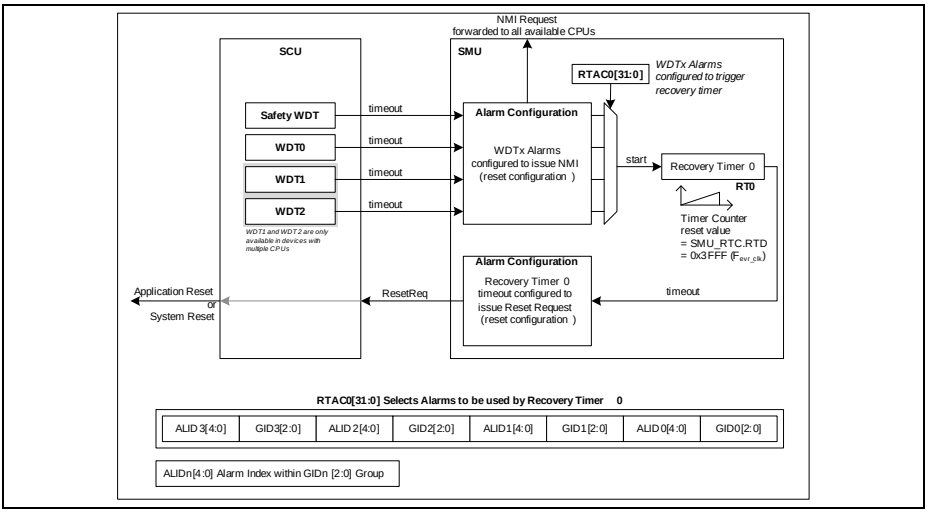


Figure 10-5 Watchdog timeout alarm configuration

The watchdog timeout detection is required from the very first instruction executed by a CPU: the SMU shall process any watchdog timeout alarm during the START state.

Note: If the same behavior is expected for all WDT alarms, it is recommended to use the global WDT timeout alarm that implements a logical OR among all WDT timeout

*alarms, thus freeing some {GID[], ALID[]} configuration pairs in **SMU_RTAC0** for other purposes.*

10.4.6 SMU Control Interface

The core functionality of the SMU is introduced through its control interface. The control interface defines how the SMU can be controlled by software, as summarized in **Table 10-26 "SMU Commands" on Page 10-48**. The control interface is directly linked to the SMU state machine (SSM) operation described in **"SMU state machine" on Page 10-50** and to the Fault Signaling Protocol (FSP) described in **"Fault Signaling Protocol (FSP)" on Page 10-52**.

The control interface is implemented by the **SMU_CMD** register using the CMD and ARG fields. The command completion status is available via the **SMU_STS** register.

Table 10-26 SMU Commands

Command	Description	Code
SMU_Start(ARG)	Forces the SSM to go to the RUN state from the START state. Argument ARG shall be set to 0.	0x0
SMU_ActivateFSP(ARG)	Activates the Fault Signaling Protocol. This action is possible in any state of the SSM. Argument ARG shall be set to 0.	0x1
SMU_ReleaseFSP(ARG)	Turns the FSP into the fault free state. In the START state, SMU_ActivateFSP() and SMU_ReleaseFSP() can be called as many times as necessary to perform self tests of every alarm source. Argument ARG shall be set to 0.	0x2
SMU_ActivatePES(ARG)	Triggers the activation of the Port Emergency Stop (PES). The PES is also directly controlled by the SMU when entering the FAULT state. Argument ARG shall be set to 0.	0x3
SMU_RTSop()	Stop the recovery Timer. Argument ARG shall be set to the recovery timer instance available in the product.	0x4

Table 10-26 SMU Commands (cont'd)

Command	Description	Code
SMU_ASCE(ARG)	Alarm Status Clear Enable Command. Software shall execute this command prior to clear a AG<n> alarm status bit. This command sets the ASCE bit in the **SMU_STS** register. Argument ARG shall be set to 0.	0x5
SMU_Alarm(ARG)	Triggers a software based alarm. ARG specifies the alarm index according to the mapping defined in **"Alarm Mapping" on Page 10-14**. A software alarm has the same properties as an hardware alarm.	0x6

Note: If the argument does not comply with the specification of the command the command is ignored and returns an error code.

The next table specifies the legal conditions that shall be followed for the execution of the commands. The conditions depend on the SMU state machine (SSM) states (see **"SMU state machine" on Page 10-50**). Any situation not specified leads to an error code.

Table 10-27 SMU commands and valid conditions

Command	SSM state	Other conditions
SMU_Start(ARG)	START	ARG == 0
SMU_ActivateFSP(ARG)	Any	ARG == 0
SMU_ReleaseFSP(ARG)	START	ARG == 0
SMU_ReleaseFSP(ARG)	FAULT	ARG == 0 & **SMU_AGC**.EFRST == 1[1)]
SMU_ActivatePES(ARG)	Any	ARG == 0
SMU_RTStop(ARG)	Any	ARG >= 0 and ARG <= Number of Recovery Timer Instances and Recovery Timer Enabled
SMU_ASCE(ARG)	Any	ARG == 0
SMU_Alarm(ARG)	RUN, FAULT[2)]	ARG >= 0

1) See also **"Fault Signaling Protocol (FSP)" on Page 10-52**.

2) In the START state of the SMU input alarms are not processed, therefore software triggered alarms will have no effect.

10.4.7 SMU state machine

Figure 10-6 "SMU state machine (SSM): transition conditions and actions" on Page 10-50 and **Figure 10-6 "SMU state machine (SSM): transition conditions and actions" on Page 10-50** fully specifies the behavior of the SMU state machine (SSM).

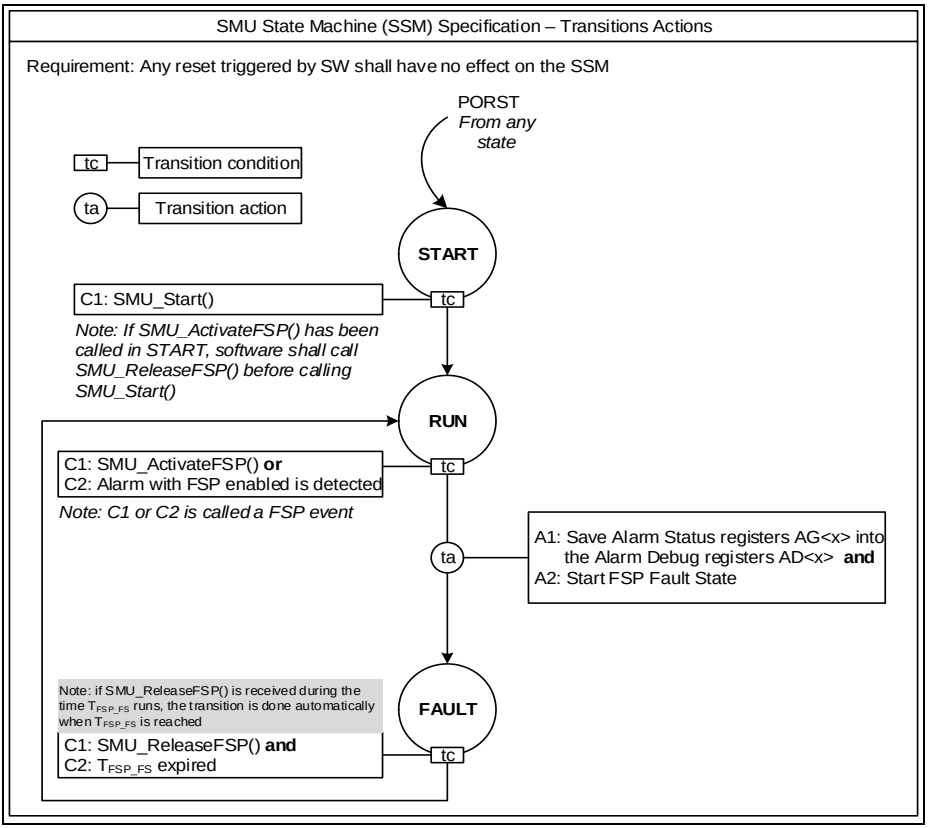


Figure 10-6 SMU state machine (SSM): transition conditions and actions

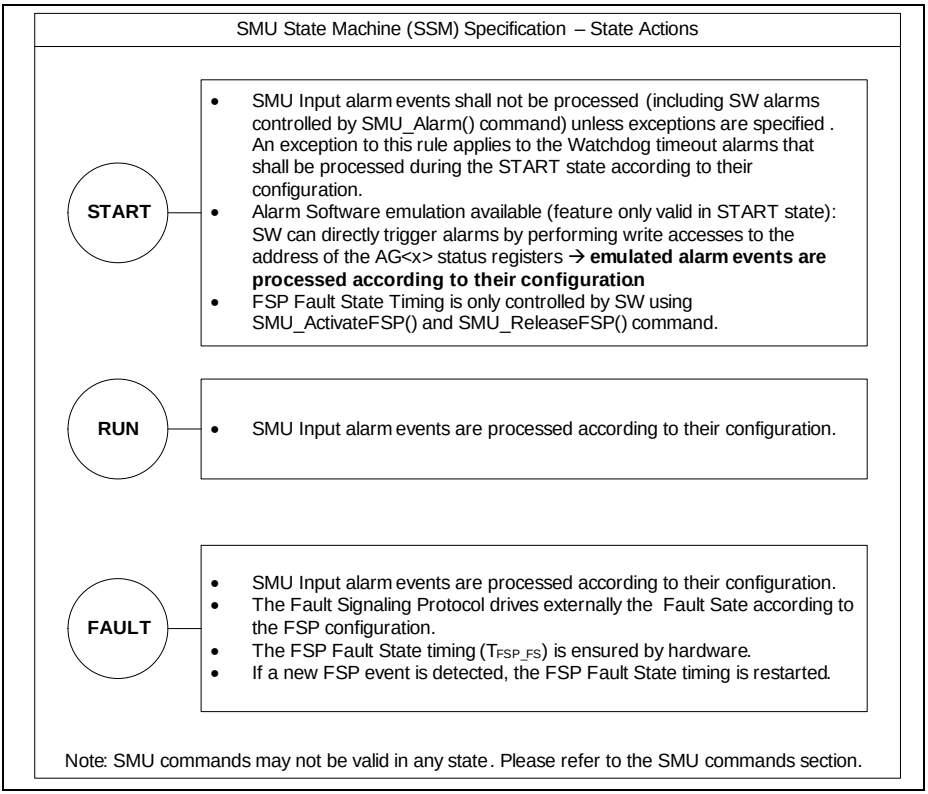


Figure 10-7 SMU state machine: state actions

Fault Counter

The SMU implements a Fault Counter (**SMU_AFCNT**) that counts the number of transitions from the RUN state to the FAULT state. The Fault Counter register is only reset by a power-on-reset.

10.4.8 Fault Signaling Protocol (FSP)

The Fault Signaling Protocol enables the microcontroller to report a critical situation to an external safety controller device in order to control the safe state of the safety system.

10.4.8.1 Introduction

The fault signaling protocol is configured via the **SMU_FSP** command register. The FSP has three states:

- The power-on reset state. After power-on reset the SMU is disconnected from the ports (see **"SMU Integration Guidelines" on Page 10-13**). After power-on reset the SMU FSP output shall be the Fault State.
- The Fault-free State. Whenever the fault-free state is controlled by a timing, the timing will be called T_{FSP_FFS} and is controlled by the **SMU_FSP** register.
- The Fault State. The timing of the fault state is controlled by the **SMU_FSP** register. The minimum active fault state time is called $T_{FSP_FS.}$

The Fault-free and fault state behavior can be configured with the following protocols:

- Bi-stable protocol (default)
- Time-switching protocol

The FSP can be controlled by:

- Software using the SMU_ActivateFSP() and SMU_ReleaseFSP() commands using the **SMU_CMD** register
- Hardware based on the **SMU_AG0FSP**, **SMU_AG1FSP**, **SMU_AG2FSP**, **SMU_AG3FSP**, **SMU_AG4FSP**, **SMU_AG5FSP**, **SMU_AG6FSP** configuration registers.
- The logic state of the FSP PAD can be observed by reading the FSP field of the **SMU_STS** register.

Figure 10-8 "Reference clocks for FSP timings" on Page 10-53 specifies the intermediate clocks to generate the T_{FSP_FFS} and T_{FSP_FS} timings.

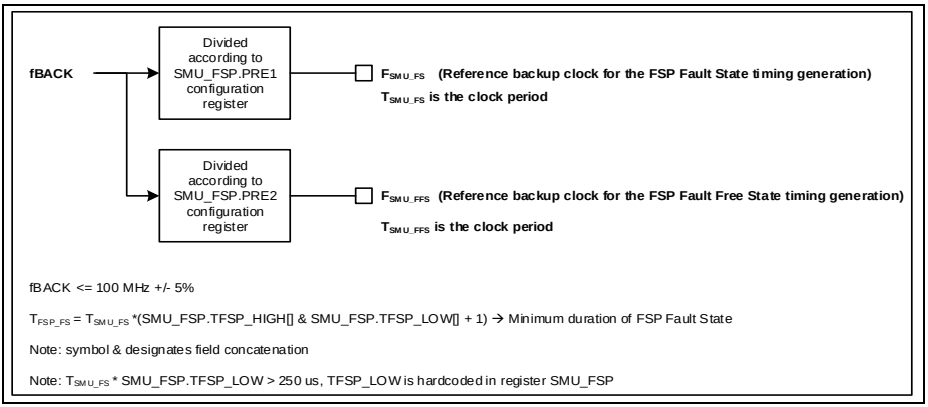


Figure 10-8 Reference clocks for FSP timings

10.4.8.2 Bi-stable fault signaling protocol

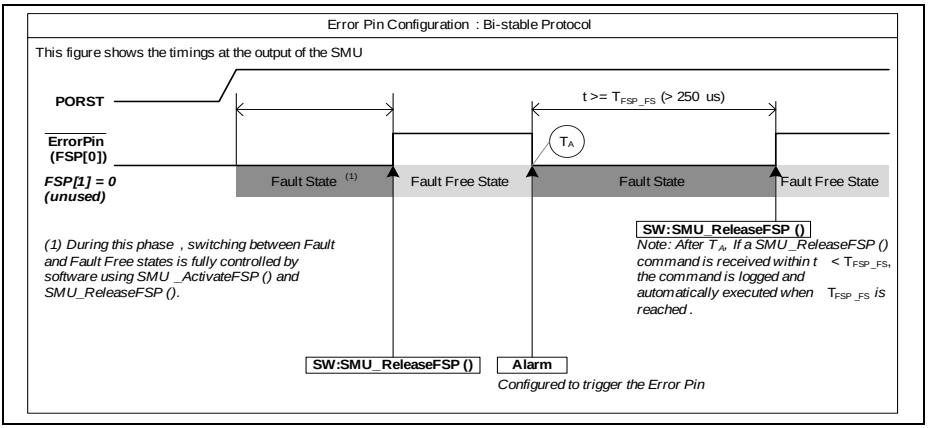


Figure 10-9 Bi-stable fault signaling protocol

Operation

FSP[0] is switched to a static logic level 0 or logic level 1 under software or hardware control.

- During power-on-reset FSP[0] = 0 (fault state)
- After power-on reset FSP[0] stays in the fault state
- FSP[0] must be set to the fault free state per software (SMU_ReleaseFSP()).
- Upon detection of an alarm event configured to activate the FSP, FSP[0] goes to the fault state and remains in this state until a SMU_ReleaseFSP() command is received and T_{FSP_FS} is satisfied or a Power-on Reset takes place.

10.4.8.3 Time switching protocol

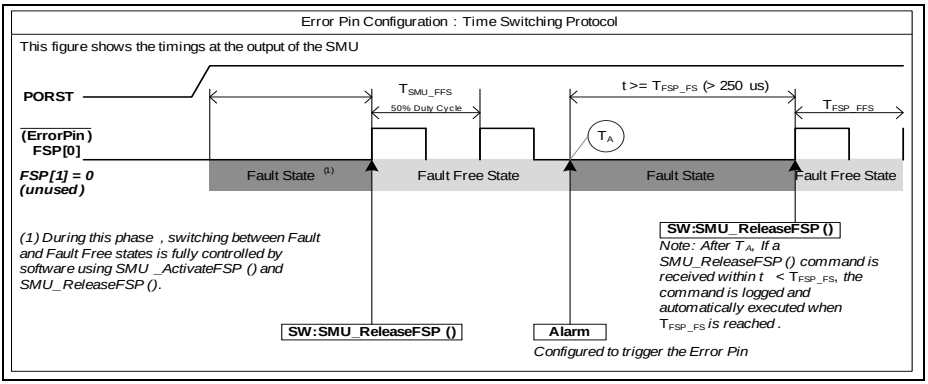


Figure 10-10 Time switching protocol

Operation

FSP[0] is toggled between logic level 0 and logic level 1 with a defined frequency. This frequency modulation protocol is violated when the SMU enters the FAULT state.

- During power-on-reset FSP[0] = 0 (fault state).
- After power-on-reset FSP[0] stays in the fault state.
- FSP[0] must be set to the fault free state per software (SMU_ReleaseFSP()).
- In the fault free state, FSP[0] oscillates between logic level 0 and logic level 1 with the frequency configured via the **SMU_FSP** register (see **Figure 10-10**).
- Upon detection of an alarm event configured to activate the FSP, FSP[0] goes immediately to the fault state and remains in this state until a SMU_ReleaseFSP() command is received and T_{FSP_FS} is satisfied or a Power-on Reset takes place.

10.4.8.4 FSP Fault State

When an alarm configured to activate the FSP, the SMU automatically switches to the FAULT state. During this time it is also possible for the safety-related software to try to analyze the root cause (when the microcontroller is still operational) and decide about the severity of the error. As the FSP is active for at least T_{FSP_FS} (see **SMU_FSP** register), it is ensured that the safe state of the system is entered through external mechanisms independent from the microcontroller (besides FSP itself).

While in the Fault State, if a new alarm event configured to activate the FSP is received and the T_{FSP_FS} has not yet been reached, the T_{FSP_FS} timing shall be restarted.

During the time TFSP_FS FSP is in the Fault State, software may have concluded that the fault is uncritical and decides to issue a SMU_ReleaseFSP() command, notifying the SMU it can return to the RUN state (the run-time of the software error handler is not directly correlated with the TFSP_FS duration and in practice shall be much shorter). This feature shall be used with caution: a microcontroller reset is highly recommended to restart the operation of the safety function when a fault reported by the SMU is assessed to be uncritical. Therefore this feature is per default disabled and shall be configured with the EFRST (Enable Fault to Run State Transition) field in the **SMU_AGC** register.

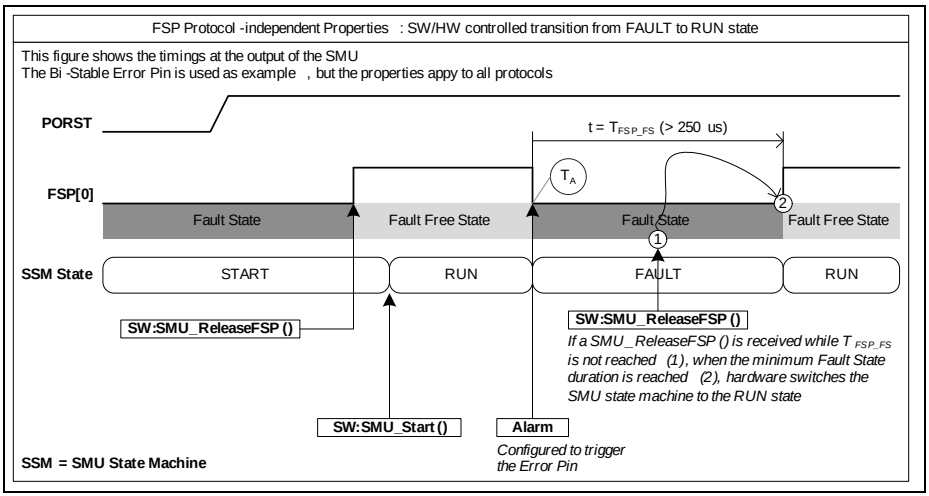


Figure 10-11 FSP: Fault State to Fault Free State Transition

10.4.8.5 FSP and SMU START State

Figure 10-12 "Software Control of FSP during SMU START State" on Page 10-57 shows a typical use case where the FSP transitions between the Fault State and Fault Free State are controlled by software using the SMU_ReleaseFSP() and SMU_ActivateFSP() commands.

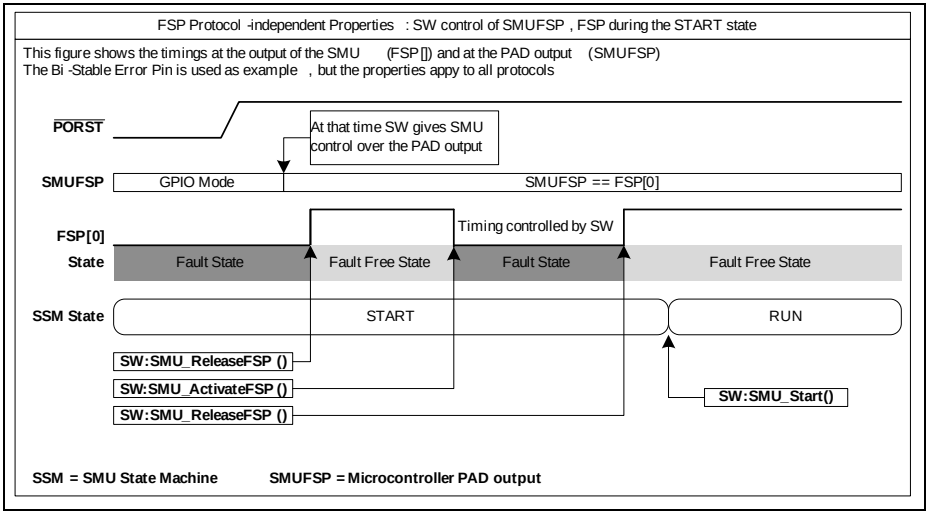


Figure 10-12 Software Control of FSP during SMU START State

Conditions of Use

- Software shall ensure that the FSP is in the Fault Free State before entering the RUN state with the SMU_Start() command.

10.4.9 OCDS Trigger Bus (OTGB) Interface

The SMU concentrates all failure indicator signals (alarms) of the device. By using them as MCDS trace information (failure type) and trace control (stop trace recording), the analysis of a failure's root cause is supported. This is even the case, when the alarm handling within the system includes a PORST, since the content of the trace memory will be still valid after the PORST.

The alarms handled in the SMU are very rare and sporadic. So it's acceptable that they are visible on the Trigger Bus with a small delay and not all in parallel with a cycle accurate timing resolution. The SMU Trigger Set is shown in **Table 10-28**. It is output on OTGB0 or OTGB1 controlled by the **SMU_OCS** register.

Table 10-28 TS16_SMU Trigger Set SMU

Bits	Name	Description
0	AA	Any ALM bit active
1		Reserved
[3:2]	ABI	ALM Byte Index, selecting the byte within the ALM group
[6:4]	AGI	ALM Group Index
7		Reserved
[15:8]	ALM	Selected byte of the ALM group

If no alarm is active, TS16_SMU will be all zero. If alarms are only active within one ALM Byte, TS16_SMU will show statically this byte. If alarms are active in two or more ALM Bytes, TS16_SMU will change between these bytes.

Note that due to the implementation, that all the ALM Bytes are being scanned one per clock cycle, the TS16_SMU ALM Byte information on OTGB0/1 can be delayed by up to 27 clock cycles for the second alarm. TS16_SMU.AA will however become active (and inactive) immediately. The implementation will make sure, that the ALM Byte with the first causing alarm will be traced with MCDS before a reset as safety action clears this information.

10.4.10 Register Properties

10.4.10.1 Register Write Protection

The SMU registers are write protected against illegal master accesses by the master protection mechanism implemented in the System Peripheral Bus interface logic. The registers controlling the master access protection are described in the section **"System Registers description" on Page 10-68**. In addition the SMU registers can only be written if the SafeEinit is enabled. Read accesses have no restriction as there is no side effect specified when reading registers. The SafeEinit has the same properties as the system Einit but it is generated by the safety watchdog. In order to access a SMU register the software must first activate the safety watchdog via a signature protected sequence. The safety watchdog is configured to enable only safety-related software to activate the SafeEinit.

Configuration Lock

In addition to the aforementioned standard features, additional mechanisms are implemented to control and protect the SMU configuration. In order to configure and lock the SMU configuration the following steps shall be followed:

- The SMU configuration is only possible if the CFGLCK field of the **SMU_KEYS** register is set to 0xBC.
- If the PERLCK field of the **SMU_KEYS** register is set to 0xFF no further SMU configuration is possible, including the **SMU_KEYS** register itself. No SMU configuration is possible anymore until the PERLCK field of the **SMU_KEYS** register is reset to 0x00 by a an application reset.

The SMU configuration registers controlled by the **SMU_KEYS** register properties are:

- **SMU_FSP**
- **SMU_AGC**
- **SMU_RTC**
- **SMU_RTAC0**
- **SMU_RTAC1**
- **SMU_AG0CFx (x=0-2)**
- **SMU_AG1CFx (x=0-2)**
- **SMU_AG2CFx (x=0-0)**
- **SMU_AG3CFx (x=0-0)**
- **SMU_AG4CFx (x=0-2)**
- **SMU_AG5CFx (x=0-2)**
- **SMU_AG6CFx (x=0-2)**
- **SMU_AG0FSP**
- **SMU_AG1FSP**

- **SMU_AG2FSP**
- **SMU_AG3FSP**
- **SMU_AG4FSP**
- **SMU_AG5FSP**
- **SMU_AG6FSP**
- **SMU_PCTL**
- **SMU_RMCTL**

The **SMU_CMD** register is not locked as it is used for run-time hardware/software interaction, it is not a configuration register.

The **SMU_AGx (x=0-6)** registers are not locked as it is possible during run-time to clear the alarm flags by software.

The **SMU_OCS**, **SMU_KRSTCLR**, **SMU_KRST1**, **SMU_KRST0**, **SMU_ACCEN1**, **SMU_ACCEN0** do not belong to the SMU kernel but to the standard bus interface module, therefore they are not controlled by the **SMU_KEYS** register.

The read only registers are not protected by the lock mechanism.

10.4.10.2 Safety Flip-Flops

Safety Flip-Flops are special Flip-Flops that implement an hardware mechanism capable to detect single event effects that may lead to single event upsets (bit flip). The SMU configuration registers that shall be implemented with safety flip-flops are:

- **SMU_FSP**
- **SMU_AGC**
- **SMU_RTC**
- **SMU_KEYS**
- **SMU_PCTL**
- **SMU_RTAC0**
- **SMU_RTAC1**
- **SMU_AG0CFx (x=0-2)**
- **SMU_AG1CFx (x=0-2)**
- **SMU_AG2CFx (x=0-0)**
- **SMU_AG3CFx (x=0-0)**
- **SMU_AG4CFx (x=0-2)**
- **SMU_AG5CFx (x=0-2)**
- **SMU_AG6CFx (x=0-2)**
- **SMU_AG0FSP**
- **SMU_AG1FSP**
- **SMU_AG2FSP**
- **SMU_AG3FSP**
- **SMU_AG4FSP**
- **SMU_AG5FSP**
- **SMU_AG6FSP**

Additionally the following SMU functions shall also be implemented with safety flip-flops:

- SMU state machine registers
- Registers implementing the FSP function

10.5 SMU Module Registers

Figure 10-13 shows the SMU module register map.

Table 10-29 shows the SMU Address Space

Table 10-30 lists all registers implemented in the SMU.

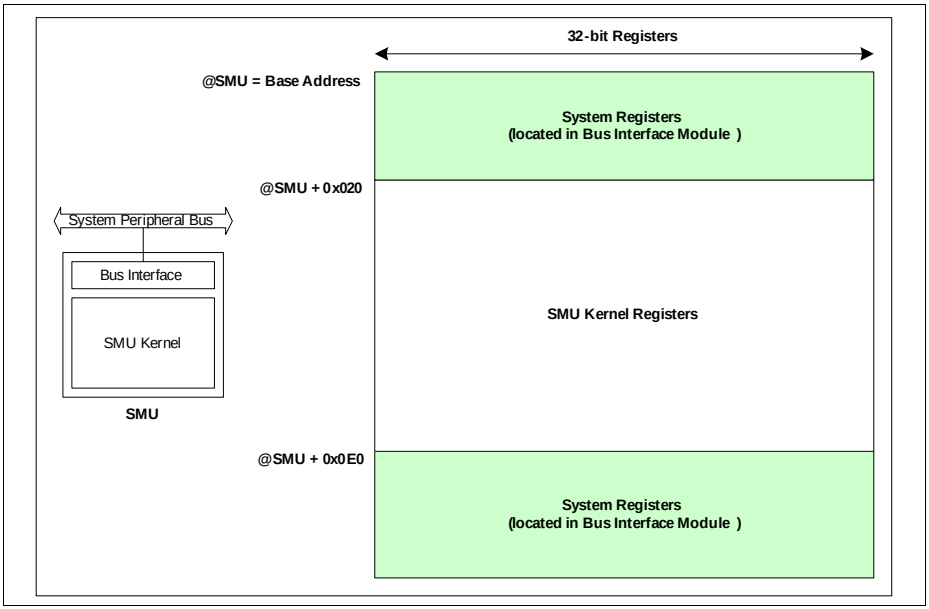


Figure 10-13 SMU Register Map

Table 10-29 Registers Address Space for TC21x/TC22x/TC23x

Module	Base Address	End Address	Note
SMU	F003 6800$_H$	F003 6FFF$_H$	Registers

Table 10-30 Registers Overview

Short Name	Description	Offset Addr	Access Mode		Reset Type	Description See
			Read	Write		
System Registers						
CLC	Clock Control Register	00_H	U, SV	SV,P	Application Reset	Page 10-68
ID	Module Identifier	08_H	U, SV	BE	Application Reset	Page 10-70
Kernel Registers:						
CMD	Command interface	20_H	U, SV	SV,P,32	Application Reset	Page 10-79
STS	Status	24_H	U, SV	SV,P,32	Application Reset	Page 10-80
FSP	FSP control	28_H	U, SV	SV,P,SE,32	Power-on Reset	Page 10-83
AGC	Alarm Global Configuration	$2C_H$	U, SV	SV,P,SE,32	Application Reset	Page 10-85
RTC	Recovery Timer Configuration	30_H	U, SV	SV,P,SE,32	Application Reset	Page 10-87
KEYS	Register access keys	34_H	U, SV	SV,P,SE,32	Application Reset	Page 10-88
DBG	Hardware debug	38_H	U, SV	BE	Power-on Reset	Page 10-89
PCTL	FSP Port Control Register	$3C_H$	U, SV	SV,P,SE,32	Power-on Reset	Page 10-90
AFCNT	Alarm and Fault Counter Register	40_H	U, SV	BE	Power-on Reset	Page 10-92
RTAC0	Recovery Timer 0 Alarm Configuration	60_H	U, SV	SV,P,SE,32	Application Reset	Page 10-93
RTAC1	Recovery Timer 1 Alarm Configuration	64_H	U, SV	SV,P,SE,32	Application Reset	Page 10-95

Table 10-30 Registers Overview (cont'd)

Short Name	Description	Offset Addr	Access Mode		Reset Type	Description See
			Read	Write		
AG0CF0	Alarm configuration	100_H	U, SV	SV,P,SE,32	Application Reset	Page 10-97
AG0CF1	Alarm configuration	104_H	U, SV	SV,P,SE,32	Application Reset	Page 10-97
AG0CF2	Alarm configuration	108_H	U, SV	SV,P,SE,32	Application Reset	Page 10-97
AG1CF0	Alarm configuration	$10C_H$	U, SV	SV,P,SE,32	Application Reset	Page 10-98
AG1CF1	Alarm configuration	110_H	U, SV	SV,P,SE,32	Application Reset	Page 10-98
AG1CF2	Alarm configuration	114_H	U, SV	SV,P,SE,32	Application Reset	Page 10-98
AG2CF0	Alarm configuration	118_H	U, SV	SV,P,SE,32	Application Reset	Page 10-99
AG2CF1	Alarm configuration	$11C_H$	U, SV	SV,P,SE,32	Application Reset	Page 10-99
AG2CF2	Alarm configuration	120_H	U, SV	SV,P,SE,32	Application Reset	Page 10-99
AG3CF0	Alarm configuration	124_H	U, SV	SV,P,SE,32	Application Reset	Page 10-102
AG3CF1	Alarm configuration	128_H	U, SV	SV,P,SE,32	Application Reset	Page 10-102
AG3CF2	Alarm Configuration	$12C_H$	U, SV	SV,P,SE,32	Application Reset	Page 10-102
AG4CF0	Alarm configuration	130_H	U, SV	SV,P,SE,32	Application Reset	Page 10-105
AG4CF1	Alarm configuration	134_H	U, SV	SV,P,SE,32	Application Reset	Page 10-105
AG4CF2	Alarm Configuration	138_H	U, SV	SV,P,SE,32	Application Reset	Page 10-105

Table 10-30 Registers Overview (cont'd)

Short Name	Description	Offset Addr	Access Mode		Reset Type	Description See
			Read	Write		
AG5CF0	Alarm configuration	$13C_H$	U, SV	SV,P,SE,32	Application Reset	Page 10-106
AG5CF1	Alarm configuration	140_H	U, SV	SV,P,SE,32	Application Reset	Page 10-106
AG5CF2	Alarm Configuration	144_H	U, SV	SV,P,SE,32	Application Reset	Page 10-106
AG6CF0	Alarm configuration	148_H	U, SV	SV,P,SE,32	Application Reset	Page 10-107
AG6CF1	Alarm configuration	$14C_H$	U, SV	SV,P,SE,32	Application Reset	Page 10-107
AG6CF2	Alarm Configuration	150_H	U, SV	SV,P,SE,32	Application Reset	Page 10-107
AG0FSP	FSP configuration for Alarm Group 0	180_H	U, SV	SV,P,SE,32	Application Reset	Page 10-108
AG1FSP	FSP configuration for Alarm Group 1	184_H	U, SV	SV,P,SE,32	Application Reset	Page 10-109
AG2FSP	FSP configuration for Alarm Group 2	188_H	U, SV	SV,P,SE,32	Application Reset	Page 10-110
AG3FSP	FSP configuration for Alarm Group 3	$18C_H$	U, SV	SV,P,SE,32	Application Reset	Page 10-111
AG4FSP	FSP configuration for Alarm Group 4	190_H	U, SV	SV,P,SE,32	Application Reset	Page 10-112
AG5FSP	FSP configuration for Alarm Group 5	194_H	U, SV	SV,P,SE,32	Application Reset	Page 10-113
AG6FSP	FSP configuration for Alarm Group 6	198_H	U, SV	SV,P,SE,32	Application Reset	Page 10-114
AG0	Alarm Group 0 Status	$1C0_H$	U, SV	SV,P,SE,32	Application Reset	Page 10-115
AG1	Alarm Group 1 Status	$1C4_H$	U, SV	SV,P,SE,32	Application Reset	Page 10-115

Table 10-30 Registers Overview (cont'd)

Short Name	Description	Offset Addr	Access Mode		Reset Type	Description See
			Read	Write		
AG2	Alarm Group 2 Status	$1C8_H$	U, SV	SV,P,SE,32	Application Reset	Page 10-115
AG3	Alarm Group 3 Status	$1CC_H$	U, SV	SV,P,SE,32	Application Reset	Page 10-115
AG4	Alarm Group 4 Status	$1D0_H$	U, SV	SV,P,SE,32	Application Reset	Page 10-115
AG5	Alarm Group 5 Status	$1D4_H$	U, SV	SV,P,SE,32	Application Reset	Page 10-115
AG6	Alarm Group 6 Status	$1D8_H$	U, SV	SV,P,SE,32	Application Reset	Page 10-115
AD0	Alarm Group 0 Debug	200_H	U, SV	BE	Power-on Reset	Page 10-116
AD1	Alarm Group 1 Debug	204_H	U, SV	BE	Power-on Reset	Page 10-116
AD2	Alarm Group 2 Debug	208_H	U, SV	BE	Power-on Reset	Page 10-116
AD3	Alarm Group 3 Debug	$20C_H$	U, SV	BE	Power-on Reset	Page 10-116
AD4	Alarm Group 4 Debug	210_H	U, SV	BE	Power-on Reset	Page 10-116
AD5	Alarm Group 5 Debug	214_H	U, SV	BE	Power-on Reset	Page 10-116
AD6	Alarm Group 6 Debug	218_H	U, SV	BE	Power-on Reset	Page 10-116
Special Safety Registers: Safety Flip-Flop Safety Mechanism						
RMCTL	Safety Flip-Flop Test Mode Control	300_H	U, SV	SV,P,SE,32	Application Reset	Page 10-117
RMEF	Safety Flip-Flop Error Flag	304_H	U, SV	SV,P,SE,32	Application Reset	Page 10-118
RMSTS	Safety Flip-Flop Self Test Status	308_H	U, SV	SV,P,SE,32	Application Reset	Page 10-118

Table 10-30 Registers Overview (cont'd)

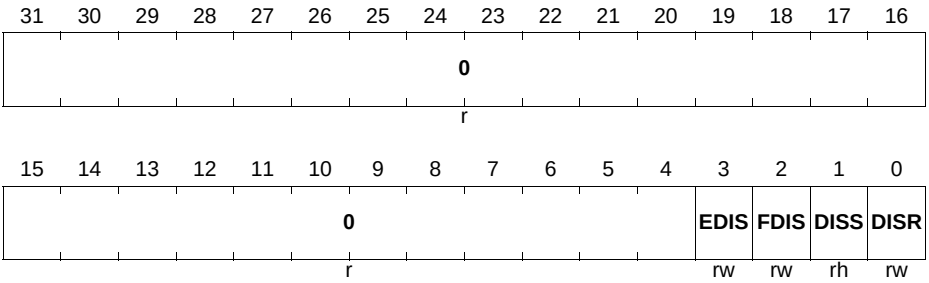
Short Name	Description	Offset Addr	Access Mode		Reset Type	Description See
			Read	Write		
System Registers						
OCS	OCDS Control and Status Register	$7E8_H$	U, SV	SV,P	Debug Reset	**Page 10-71**
KRSTCLR	Reset Status Clear Register	$7EC_H$	U, SV	SV,P	Application Reset	**Page 10-73**
KRST1	Reset Control Register 1	$7F0_H$	U, SV	SV,P	Application Reset	**Page 10-74**
KRST0	Reset Control Register 0	$7F4_H$	U, SV	SV,P	Application Reset	**Page 10-75**
ACCEN1	Access Enable Register	$7F8_H$	U, SV	BE	Application Reset	**Page 10-77**
ACCEN0	Access Enable Register	$7FC_H$	U, SV	SV,SE	Application Reset	**Page 10-78**

10.5.1 System Registers description

SMU Clock Control Register (CLC)

The Clock Control Register allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The SMU shall be enabled per default

SMU_CLC
Clock Control Register **(00_H)** **Reset Value: 0000 0000_H**



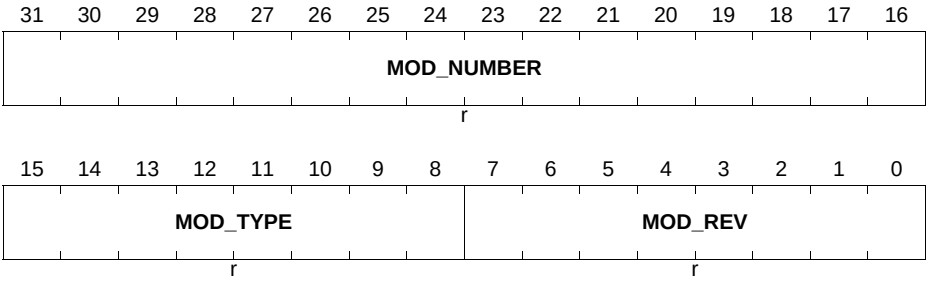
Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. 0_B Module disable is not requested 1_B Module disable is requested
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module. 0_B Module is enabled 1_B Module is disabled
FDIS	2	rw	**Force Disable** The feature controlled by this field is not used. Read as 0; should be written with 0.
EDIS	3	rw	**Sleep Mode Enable Control** Used to control module's sleep mode. Sleep Mode is not supported safety applications. During the process of entering and resuming from sleep mode the intended processing of alarm events is not guaranteed.

Field	Bits	Type	Description
0	[31:4]	r	**Reserved** Read as 0; should be written with 0.

Note: The other features controlled by the CLC register are not supported by the SMU.

SMU Module Identification Register

SMU_ID
Module Identification Register **(08_H)** **Reset Value: $0089\ C001_H$**



Field	Bits	Type	Description
MOD_REV	[7:0]	r	**Module Revision Number** This bit field defines the module revision number. The value of a module revision starts with 01_H (first revision).
MOD_TYPE	[15:8]	r	**Module Type** The bit field is set to $C0_H$ which defines the module as a 32-bit module.
MOD_NUMBER	[31:16]	r	**Module Number Value** This bit field defines a module identification number. The value for the SMU module is 0089_H.

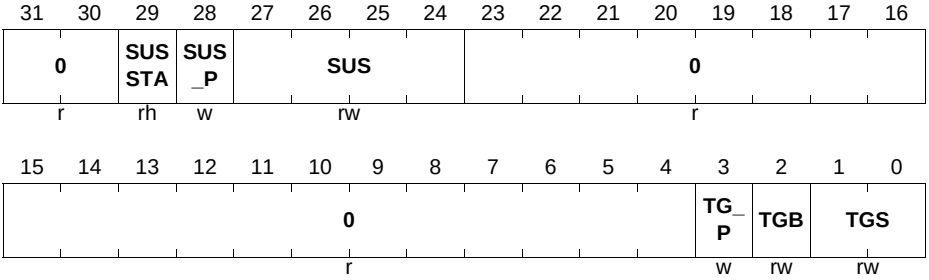
SMU OCDS Control Register. This register is implemented in the BPI.

The OCDS Control and Status (OCS) register is reset by Debug Reset.

The OCS register includes the module related control bits for the OCDS Trigger Bus (OTGB).

The register can only be written and the OCS control register bits are only effective while the OCDS is enabled (OCDS enable = ´1´). While OCDS is not enabled, OCS reset values/modes are effective.

SMU_OCS
OCDS Control and Status **($7E8_H$)** **Reset Value: $0000\ 0000_H$**

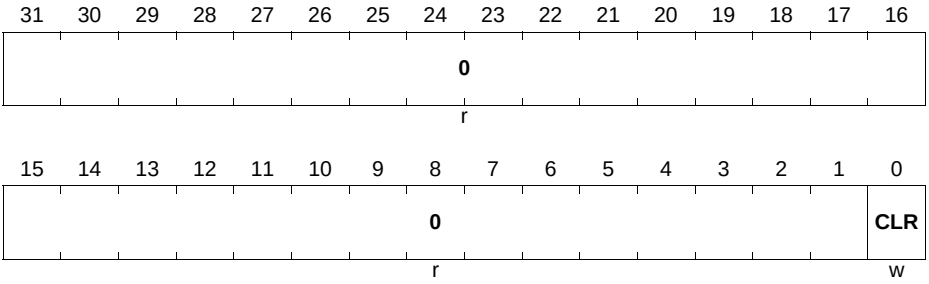


Field	Bits	Type	Description
TGS	[1:0]	rw	**Trigger Set for OTGB0/1** 0_H No Trigger Set output 1_H TS16_SMU **others**, reserved
TGB	2	rw	**OTGB0/1 Bus Select** 0_B Trigger Set is output on OTGB0 1_B Trigger Set is output on OTGB1
TG_P	3	w	**TGS, TGB Write Protection** TGS and TGB are only written when TG_P is 1, otherwise unchanged. Read as 0.
SUS	[27:24]	rw	**OCDS Suspend Control** No effect. SMU will not suspend. Read as 0; must be written with 0.
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.

Field	Bits	Type	Description
SUSSTA	29	rh	**Suspend State** Read as 0; must be written with 0.
0	[23:4], [31:30]	r	**Reserved** Read as 0; must be written with 0.

The Kernel Reset Register Clear register is used to clear the Kernel Reset Status bit (<>_KRST0.RSTSTAT).

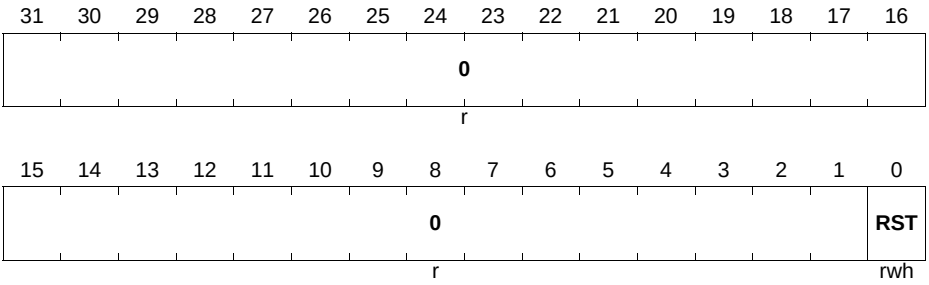
SMU_KRSTCLR
SMU Reset Status Clear Register (7EC$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT Read always as 0.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

The Kernel Reset Register 1 is used to reset the related module kernel. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers (<>_KRSTx1.RST and <>_KRSTx0.RST) related to the module kernel that should be reset (kernel 0 or kernel 1). The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

SMU_KRST1
SMU Reset Register 1 **($7F0_H$)** **Reset Value: 0000 0000_H**

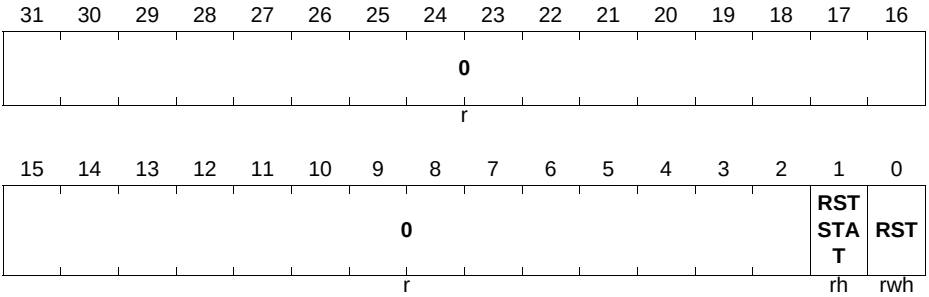


Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers related to the module kernel that should be reset (kernel 0 or kernel 1). In order support modules with two kernel the BPI_FPI provides two set of kernel reset registers. The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the related KRSTCLRx.CLR register bit.

SMU_KRST0
SMU Reset Register 0 **(7F4$_H$)** **Reset Value: 0000 0000$_H$**

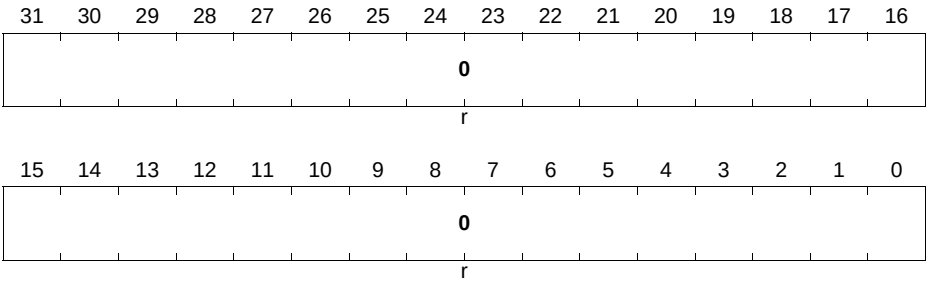


Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed.

Field	Bits	Type	Description
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. 0_B No kernel reset was executed 1_B Kernel reset was executed This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register.
0	[31:2]	r	**Reserved** Read as 0; should be written with 0.

The Access Enable Register 1 controls write access for transactions with the on chip bus master TAG ID 100000_B to 111111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the AURIX devices.

SMU_ACCEN1
SMU Access Enable Register 1 ($7F8_H$) Reset Value: 0000 0000_H



Field	Bits	Type	Description
0	[31:0]	r	**Reserved** Read as 0; should be written with 0.

The Access Enable Register 0 controls write access for transactions with the on chip bus master TAG ID 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

SMU_ACCEN0
SMU Access Enable Register 0 ($7FC_H$) Reset Value: FFFF FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	EN30	EN29	EN28	EN27	EN26	EN25	EN24	EN23	EN22	EN21	EN20	EN19	EN18	EN17	EN16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

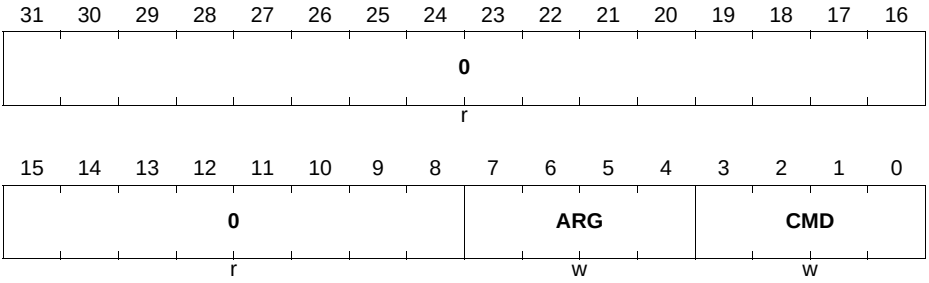
Field	Bits	Type	Description
ENn (n = 0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Note: The mapping with the on-chip master-capable modules is described in the bus chapters of the TC21x/TC22x/TC23x microcontroller.

10.5.2 SMU Configuration Registers

SMU Command Register.

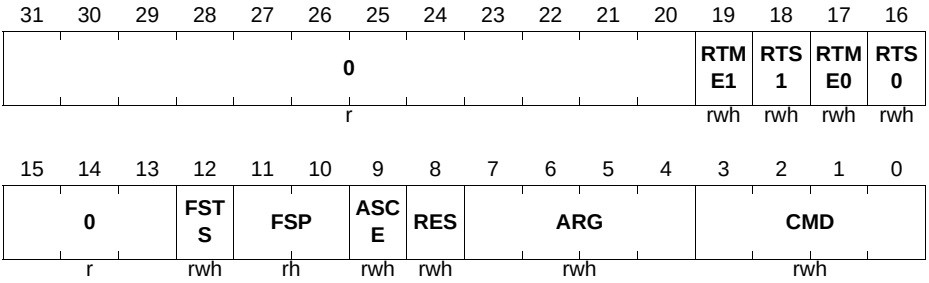
SMU_CMD
Command Register **(20_H)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CMD	[3:0]	w	**Implements the SMU Command Interface.** See **Table 10-26 “SMU Commands” on Page 10-48** for the command encoding. Read as 0.
ARG	[7:4]	w	**Implements the SMU Command Interface.** Argument to be used with the command. See **Table 10-26 “SMU Commands” on Page 10-48** for the argument encoding. Read as 0.
0	[31:8]	r	**Reserved** Read as 0; should be written with 0

SMU Status Register.

SMU_STS
Status Register **(24_H)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CMD	[3:0]	rwh	**Last command received** Same encoding as CMD field of **SMU_CMD** register
ARG	[7:4]	rwh	**Last command argument received** Same encoding as ARG field of **SMU_CMD** register
RES	8	rwh	**Result of last received command** 0_B Command was successful 1_B Command failed
ASCE	9	rwh	**Alarm Status Clear Enable** This bit controls if a status flag set in an AG<x> register upon detection of an alarm event can be cleared by software or not. When ASCE is enabled software shall write a 1 to the bit position in AG<x> to clear the bit (W1C). When a W1C action takes place the ASCE bit is automatically cleared to 0 by hardware and software shall set the ASCE bit again by using the SMU_ASCE() command. 0_B Alarm status bits AG<x> can not be cleared 1_B Alarm status bits AG<x> can be cleared.

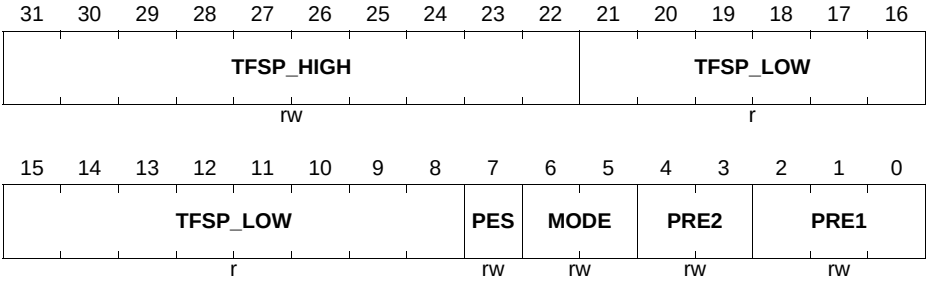
Field	Bits	Type	Description
FSP	[11:10]	rh	**Fault Signaling Protocol status** FSP[0] = FSP_STS[0] input signal FSP[1] = FSP_STS[1] input signal This field is updated by hardware every clock cycle, therefore a software clear on write is not meaningful for this field.
FSTS	12	rwh	**Fault State Timing Status** This bit indicates if the minimum timing duration of the FSP fault state has been reached or not. The bit is cleared by hardware when the fault state is entered. 0_B Minimum timing duration not reached 1_B Minimum timing duration reached
RTS0	16	rwh	**Recovery Timer 0 Status** See **"Recovery Timer" on Page 10-45** for the usage of this field. 0_B Recovery Timer not running 1_B Recovery Timer running
RTME0	17	rwh	**Recovery Timer 0 Missed Event** See **"Recovery Timer" on Page 10-45** for the usage of this field. 0_B Recovery Timer event not detected 1_B Recovery Timer event detected
RTS1	18	rwh	**Recovery Timer 1 Status** See **"Recovery Timer" on Page 10-45** for the usage of this field. 0_B Recovery Timer not running 1_B Recovery Timer running
RTME1	19	rwh	**Recovery Timer 1 Missed Event** See **"Recovery Timer" on Page 10-45** for the usage of this field. 0_B Recovery Timer event not detected 1_B Recovery Timer event detected
0	[31:20], [15:13]	r	**Reserved** Read as 0; should be written with 0.

Note: A write to this register (regardless of the data written) clears all the fields with the wrh type. If on the same cycle a software write event and hardware event is detected the hardware action wins.

SMU Fault Signalling Timing Configuration.

This register controls the timing of the fault signaling protocol.

SMU_FSP
Fault Signaling Protocol **(28_H)** **Reset Value: 003F FF00$_H$**



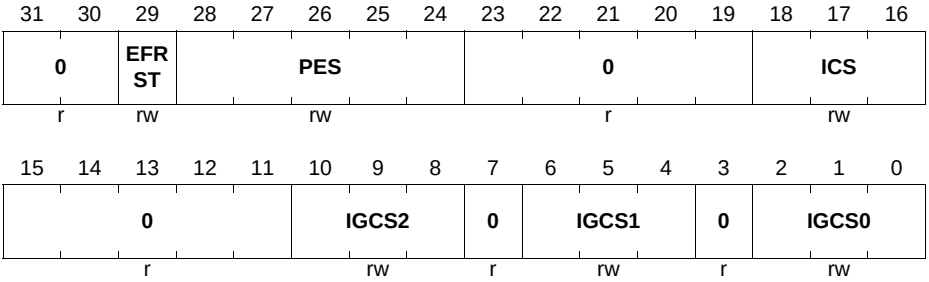
Field	Bits	Type	Description
PRE1	[2:0]	rw	**Prescaler1** Dividing factor to apply to the reference clock fBACK. **It is assumed that the maximal value for fBACK is 100 Mhz with a precision of 5%**. The divided clock is used as reference to generate the timing of the fault signaling protocol fault state. The frequency of the divided clock (called F_{SMU_FS}) is defined as follows: 0_H reference clock frequency divided by 2 1_H reference clock frequency divided by 4 2_H reference clock frequency divided by 8 3_H reference clock frequency divided by 16 4_H reference clock frequency divided by 32 5_H reference clock frequency divided by 64 6_H reference clock frequency divided by 128 7_H reference clock frequency divided by 256

Field	Bits	Type	Description
PRE2	[4:3]	rw	**Prescaler2** Dividing factor to apply to the reference clock fBACK in order to generate the timing of the fault free state for the time switching modes of the fault signaling protocol. The frequency of the divided clock (called F_{SMU_FFS}) is defined as follows: 0_H reference clock frequency divided by 512 1_H reference clock frequency divided by 1024 2_H reference clock frequency divided by 2048 3_H reference clock frequency divided by 4096
MODE	[6:5]	rw	**Fault Signaling Protocol configuration** 0_H Bi-stable protocol 1_H Reserved 2_H Time switching protocol 3_H Reserved
PES	7	rw	**Port Emergency Stop (PES)** When this bit is set a Port Emergency Stop is automatically requested when an alarm event configured to start the Fault Signalling Protocol is detected. 0_B Port Emergency Stop disabled 1_B Port Emergency Stop enabled
TFSP_LOW	[21:8]	r	**Specification of the FSP fault state duration** T_{FSP_FS}= TFSP_HIGH & TPSP_LOW. Note: symbol & designates a concatenation TFSP_LOW shall be specified as a number of FSMU_FS ticks. TFSP_LOW is defined so that the minimum duration is greater than 250 us. It can not be changed by software. Refer to **Figure 10-8 “Reference clocks for FSP timings” on Page 10-53**
TFSP_HIGH	[31:22]	rw	**Specification the FSP fault state duration** T_{FSP_FS}= TFSP_HIGH & TPSP_LOW. Note: symbol & designates a concatenation TFSP_HIGH shall be specified as a number of FSMU_FS ticks. TFSP_HIGH and PRE1 shall enable to configure a fault state duration of 500 ms.

SMU Alarm Global Configuration.

This register controls some properties related to the behavior of the SMU to alarm.

SMU_AGC
Alarm Global Configuration **($2C_H$)** **Reset Value: 0000 0000$_H$**



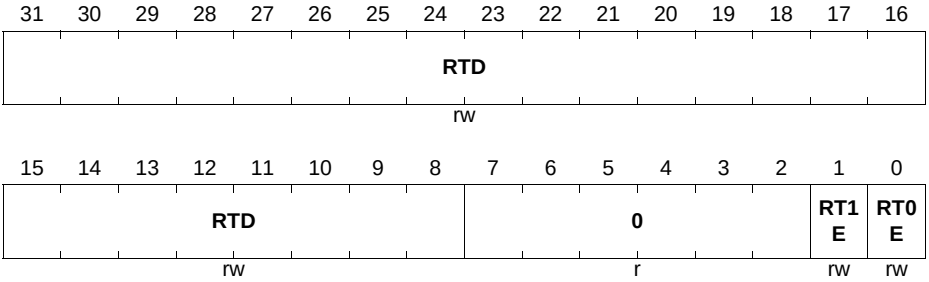
Field	Bits	Type	Description
IGCS0	[2:0]	rw	**Interrupt Generation Configuration Set 0** Defines the output value of the interrupt request vector when the alarm configuration flag selects the interrupt configuration set 0. Enables to issue an interrupt request to several CPUs: see **“Interfaces to the Interrupt Router” on Page 10-8**.
IGCS1	[6:4]	rw	**Interrupt Generation Configuration Set 1** Defines the output value of the interrupt request vector when the alarm configuration flag selects the interrupt configuration set 1. Enables to issue an interrupt request to several CPUs: see **“Interfaces to the Interrupt Router” on Page 10-8**.
IGCS2	[10:8]	rw	**Interrupt Generation Configuration Set 2** Defines the output value of the interrupt request vector when the alarm configuration flag selects the interrupt configuration set 2. Enables to issue an interrupt request to several CPUs: see **“Interfaces to the Interrupt Router” on Page 10-8**.

Field	Bits	Type	Description
ICS	[18:16]	rw	**Idle Configuration Set** Defines the output value of the idle request vector when the alarm configuration flag selects the Idle Configuration Set. Enables to issue an idle request to several CPUs if required. More complex idle scenarios can be handled by using software interrupts.
PES	[28:24]	rw	**Port Emergency Stop** This field enables control of the Port Emergency Stop (PES) feature independently for each internal action. When an action is triggered and if the corresponding bit (as defined below) is set, the hardware triggers automatically a port emergency stop request. Each bit of PES is allocated to an action as follows: 1_H SMU_IGCS0 activates PES 2_H SMU_IGCS1 activates PES 4_H SMU_iGCS2 activates PES 8_H SMU_NMI activates PES 10_H SMU_IDLE activates PES
EFRST	29	rw	**Enable FAULT to RUN State Transition** See **“FSP Fault State” on Page 10-56** chapter for the usage of this field.
0	[31:30], [23:19], [15:11], 7, 3	r	**Reserved** Read as 0; should be written with 0

SMU Recovery Timer Duration Configuration.

This register controls the timing duration of the recovery timer.

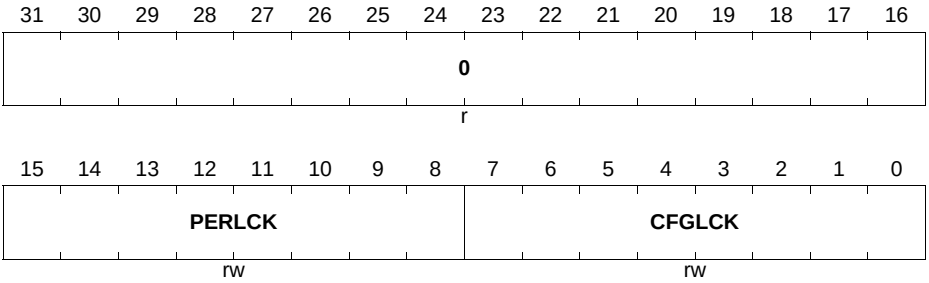
SMU_RTC
Recovery Timer Configuration (30_H) Reset Value: 003F FF01$_H$



Field	Bits	Type	Description
RTD	[31:8]	rw	**Recovery Timer Duration** This field specifies the maximum duration of the recovery timer. When the timer counter reaches the programmed value, the internal alarm rt_timeout is issued. The timer is stopped by a SMU_RTStop() command before the recovery timer. RTD shall be specified as a number of the **F_{SMU_FS} clock** ticks.
RT1E	1	rw	**RT1 Enable Bit** 0_B Recovery Timer 1 is disabled 1_B Recovery Timer 1 is enabled
RT0E	0	rw	**RT0 Enable Bit** 0_B Recovery Timer 0 is disabled 1_B Recovery Timer 0 is enabled
0	[7:2]	r	**Reserved** Read as 0; should be written with 0

SMU Keys Register.

SMU_KEYS
Key Register **(34_H)** **Reset Value: $0000\ 0000_H$**

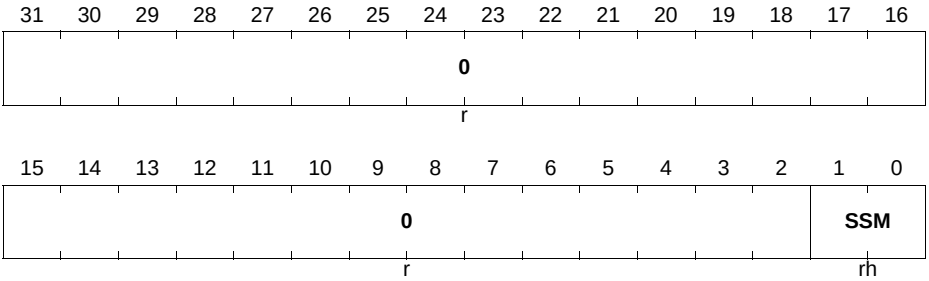


Field	Bits	Type	Description
CFGLCK	[7:0]	rw	**Configuration Lock** The SMU configuration is only possible if this field is set to 0xBC. Refer to **"Register Properties" on Page 10-59** for the list of registers controlled by this field.
PERLCK	[15:8]	rw	**Permanent Lock** If this field is set to 0xFF, no further configuration of the SMU is possible. Refer to **"Register Properties" on Page 10-59** for the list of registers controlled by this field.
0	[31:16]	r	**Reserved** Read as 0; shall be written with 0

SMU Debug Register.

This register enables to observe some internal states of the SMU hardware. Definition will be completed based on design information.

SMU_DBG
Debug Register **(38_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
SSM	[1:0]	rh	**Running state of the SMU State Machine** 0_H START state 1_H RUN state 2_H FAULT state 3_H unspecified state
0	[31:2]	r	**Reserved** Read as 0; should be written with 0

SMU register controlling the connectivity with the Ports.

SMU_PCTL
Port Control **($3C_H$)** **Reset Value: XXXX[1] 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCFG															
rh															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								PCS	0					HWEN	HWDIR
r								rw	r					rw	rw

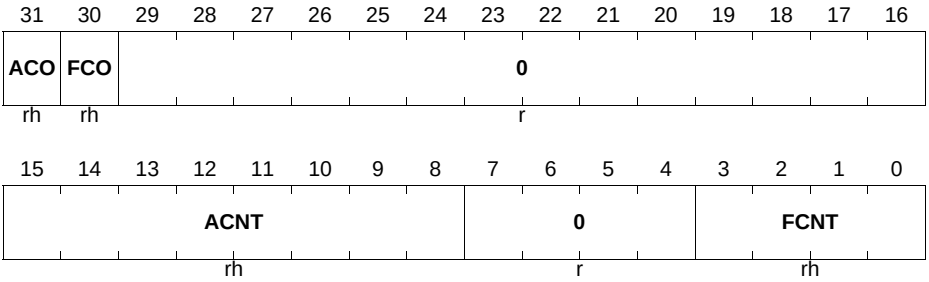
1) The PCFG reset value depends on several conditions. The PFCG value is only relevant after user software configuration.

Field	Bits	Type	Description
HWDIR	0	rw	**Port Direction.** This bit directly controls the value of the FSP_DIR SMU output signal. Also refer to the General Purpose I/O Ports chapter for the HW_DIR signal specification. 0_B sets the PAD to input state 1_B sets the PAD to output state
HWEN	1	rw	**Port Enable.** This bit directly controls the value of the FSP_EN SMU output signal. When set to 1 the port output is directly driven by SMU FSP signal (Error Pin). Also refer to the General Purpose I/O Ports chapter for the HW_EN signal specification.

Field	Bits	Type	Description
PCS	7	rw	**PAD Configuration Select** This bit controls the latching of the SMU FSP (Error Pin) PAD configuration signals to ensure that upon an application reset or system reset the SMU FSP (Error Pin) PAD configuration is not affected. This field is only reset by power-on reset. 0_B The PAD configuration is controlled by the PORT registers. 1_B The PAD configuration is controlled by the SMU. • **Only with the first transition from 0 to 1 of this field the SMU FSP is operational. Any further configuration change in this bit field has no effect to the hardware**. • **The fields HWDIR, HWEN and PCS shall be configured with a single software write command. Configuring each bit-field separately may lead to configuration inconsistencies. Refer to "Interface to the Ports (Error Pin)" on Page 10-8 for the overview of the SMU FSP (Error Pin) connectivity.**
PCFG	[31:16]	rh	**PAD Configuration** This field contains the FSP PAD configuration input signals. The PAD configuration related to FSP[0] is provided by PCFG[7:0] The upper-bits PCFG[15:8] are not used.
0	[15:8], [6:2]	r	**Reserved** Read as 0; should be written with 0

SMU Alarm and Fault Counter Register.

SMU_AFCNT
Alarm and Fault Counter **(40_H)** **Reset Value: $0000\ 0000_H$**

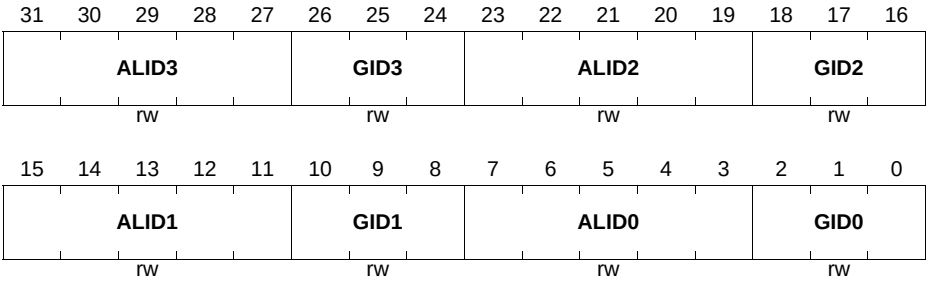


Field	Bits	Type	Description
FCNT	[3:0]	rh	**Fault Counter.** This field is incremented by hardware when the SMU state machine goes from the RUN state to the FAULT state (see **Figure 10.4.7 “SMU state machine” on Page 10-50**). The counter value holds if the maximum value is reached.
ACNT	[15:8]	rh	**Alarm Counter.** This field is incremented by hardware when the SMU processes an **internal** action related to an alarm event (see **Figure 10.4.5.3 “Alarm operation” on Page 10-43**). The counter value holds if the maximum value is reached.
FCO	30	rh	**Fault Counter Overflow.** This bit is set by hardware if the FCNT counter reached the maximum value and an increment condition is present.
ACO	31	rh	**Alarm Counter Overflow.** This bit is set by hardware if the ACNT counter reached the maximum value and an increment condition is present.
0	[29:16], [7:4]	r	**Reserved** Read as 0; should be written with 0.

SMU Recovery Timer 0 Alarm Configuration.

SMU_RTAC0

Recovery Timer Alarm Configuration (60_H) **Reset Value: A39B 938B$_H$**



Field	Bits	Type	Description
GID0	[2:0]	rw	**Group Index 0.** This field enables to specify if an alarm from this alarm group can use the recovery timer 0. The functionality of this field is described in **"Recovery Timer" on Page 10-45**.
ALID0	[7:3]	rw	**Alarm Identifier 0.** This field specifies the Alarm Index related to the Group Index specified in GID0.
GID1	[10:8]	rw	**Group Index 1.** This field enables to specify if an alarm from this alarm group can use the recovery timer 0. The functionality of this field is described in **"Recovery Timer" on Page 10-45**.
ALID1	[15:11]	rw	**Alarm Identifier 1.** This field specifies the Alarm Index related to the Group Index specified in GID1.
GID2	[18:16]	rw	**Group Index 2.** This field enables to specify if an alarm from this alarm group can use the recovery timer 0. The functionality of this field is described in **"Recovery Timer" on Page 10-45**.

Field	Bits	Type	Description
ALID2	[23:19]	rw	**Alarm Identifier 2.** This field specifies the Alarm Index related to the Group Index specified in GID2.
GID3	[26:24]	rw	**Group Index 3.** This field enables to specify if an alarm from this alarm group can use the recovery timer 0. The functionality of this field is described in **“Recovery Timer” on Page 10-45**.
ALID3	[31:27]	rw	**Alarm Identifier 3.** This field specifies the Alarm Index related to the Group Index specified in GID3.

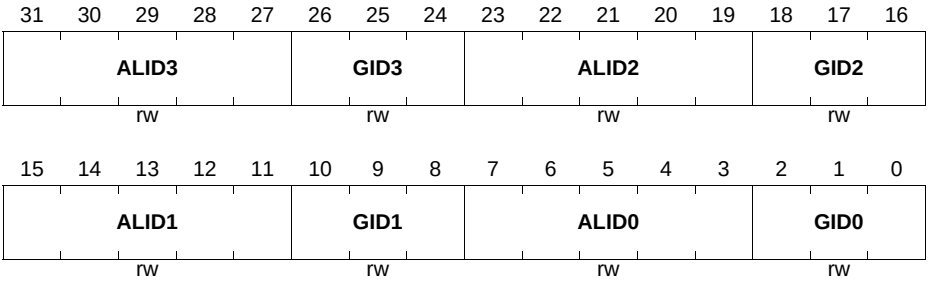
Note: It is possible to configure multiple times the same group identifier in GID0/1/2/3 fields. The reset value maps the watchdog timeout alarms to the recovery timer 0. The number of available watchdogs is product dependent.

Note: The reset values corresponds to Alarms 17,18,19,20 from Alarm Group 3

SMU Recovery Timer 1 Alarm Configuration.

SMU_RTAC1

Recovery Timer Alarm Configuration (64_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
GID0	[2:0]	rw	**Group Index 0.** This field enables to specify if an alarm from this alarm group can use the recovery timer 1. The functionality of this field is described in **"Recovery Timer" on Page 10-45**.
ALID0	[7:3]	rw	**Alarm Identifier 0.** This field specifies the Alarm Index related to the Group Index specified in GID0.
GID1	[10:8]	rw	**Group Index 1.** This field enables to specify if an alarm from this alarm group can use the recovery timer 1. The functionality of this field is described in **"Recovery Timer" on Page 10-45**.
ALID1	[15:11]	rw	**Alarm Identifier 1.** This field specifies the Alarm Index related to the Group Index specified in GID1.
GID2	[18:16]	rw	**Group Index 2.** This field enables to specify if an alarm from this alarm group can use the recovery timer 1. The functionality of this field is described in **"Recovery Timer" on Page 10-45**.

Field	Bits	Type	Description
ALID2	[23:19]	rw	**Alarm Identifier 2.** This field specifies the Alarm Index related to the Group Index specified in GID2.
GID3	[26:24]	rw	**Group Index 3.** This field enables to specify if an alarm from this alarm group can use the recovery timer 1. The functionality of this field is described in **“Recovery Timer” on Page 10-45**.
ALID3	[31:27]	rw	**Alarm Identifier 3.** This field specifies the Alarm Index related to the Group Index specified in GID3.

Note: It is possible to configure multiple times the same group identifier in GID0/1/2/3 fields.

10.5.3 SMU Alarm Configuration Registers

SMU Alarm Configuration Registers related to Alarm Group 0.

SMU_AG0CFx (x=0-2)
Alarm Configuration Register (100_H + x*04_H) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CF31	**CF30**	**CF29**	**CF28**	**CF27**	**CF26**	**CF25**	**CF24**	**CF23**	**CF22**	**CF21**	**CF20**	**CF19**	**CF18**	**CF17**	**CF16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CF15	**CF14**	**CF13**	**CF12**	**CF11**	**CF10**	**CF9**	**CF8**	**CF7**	**CF6**	**CF5**	**CF4**	**CF3**	**CF2**	**CF1**	**CF0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CFn (n = 0-31)	n	rw	**Configuration flag x (x=0-2) for alarm n belonging to alarm group 0.** The configuration flags 0, 1 and 2 must be used together to define the behavior of the SMU when a fault state is reported by the alarm n belonging to this group (see **“Alarm Configuration” on Page 10-41**). *Note:* ***If the alarm is specified as reserved in “Alarm Signals” on Page 10-24, software shall configure the behavior to “No Action”.*** 0_B Configuration flag x (x=0-2) is set to 0 1_B Configuration flag x (x=0-2) is set to 1

SMU Alarm Configuration Registers related to Alarm Group 1.

SMU_AG1CFx (x=0-2)
Alarm Configuration Register ($10C_H$ + x*04_H) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CF31	**CF30**	**CF29**	**CF28**	**CF27**	**CF26**	**CF25**	**CF24**	**CF23**	**CF22**	**CF21**	**CF20**	**CF19**	**CF18**	**CF17**	**CF16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CF15	**CF14**	**CF13**	**CF12**	**CF11**	**CF10**	**CF9**	**CF8**	**CF7**	**CF6**	**CF5**	**CF4**	**CF3**	**CF2**	**CF1**	**CF0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CFn (n = 0-31)	n	rw	**Configuration flag x (x=0-2) for alarm n belonging to alarm group 1.** The configuration flags 0, 1 and 2 must be used together to define the behavior of the SMU when a fault state is reported by the alarm n belonging to this group (see **“Alarm Configuration” on Page 10-41**). *Note: **If the alarm is specified as reserved in “Alarm Signals” on Page 10-24, software shall configure the behavior to “No Action”.*** 0_B Configuration flag x (x=0-2) is set to 0 1_B Configuration flag x (x=0-2) is set to 1

SMU Alarm Configuration Registers related to Alarm Group 2.

SMU_AG2CFx (x=0-0)
Alarm Configuration Register (118_H + x*04_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CF31	**CF30**	**CF29**	**CF28**	**CF27**	**CF26**	**CF25**	**CF24**	**CF23**	**CF22**	**CF21**	**CF20**	**CF19**	**CF18**	**CF17**	**CF16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CF15	**CF14**	**CF13**	**CF12**	**CF11**	**CF10**	**CF9**	**CF8**	**CF7**	**CF6**	**CF5**	**CF4**	**CF3**	**CF2**	**CF1**	**CF0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CFn (n = 0-31)	n	rw	**Configuration flag x (x=0-2) for alarm n belonging to alarm group 2.** The configuration flags 0, 1 and 2 must be used together to define the behavior of the SMU when a fault state is reported by the alarm n belonging to this group (see **“Alarm Configuration” on Page 10-41**). *Note: **If the alarm is specified as reserved in “Alarm Signals” on Page 10-24, software shall configure the behavior to “No Action”.*** 0_B Configuration flag x (x=0-2) is set to 0 1_B Configuration flag x (x=0-2) is set to 1

SMU Alarm Configuration Registers related to Alarm Group 2.

SMU_AG2CFx (x=1-1)
Alarm Configuration Register (118_H + x*04_H) Reset Value: 2000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CF31	**CF30**	**CF29**	**CF28**	**CF27**	**CF26**	**CF25**	**CF24**	**CF23**	**CF22**	**CF21**	**CF20**	**CF19**	**CF18**	**CF17**	**CF16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CF15	**CF14**	**CF13**	**CF12**	**CF11**	**CF10**	**CF9**	**CF8**	**CF7**	**CF6**	**CF5**	**CF4**	**CF3**	**CF2**	**CF1**	**CF0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CFn (n = 0-31)	n	rw	**Configuration flag x (x=0-2) for alarm n belonging to alarm group 2.** The configuration flags 0, 1 and 2 must be used together to define the behavior of the SMU when a fault state is reported by the alarm n belonging to this group (see **“Alarm Configuration” on Page 10-41**). *Note: **If the alarm is specified as reserved in “Alarm Signals” on Page 10-24, software shall configure the behavior to “No Action”.*** 0_B Configuration flag x (x=0-2) is set to 0 1_B Configuration flag x (x=0-2) is set to 1

SMU Alarm Configuration Registers related to Alarm Group 2.

SMU_AG2CFx (x=2-2)
Alarm Configuration Register (118_H + x*04_H) Reset Value: 2000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CF31	**CF30**	**CF29**	**CF28**	**CF27**	**CF26**	**CF25**	**CF24**	**CF23**	**CF22**	**CF21**	**CF20**	**CF19**	**CF18**	**CF17**	**CF16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CF15	**CF14**	**CF13**	**CF12**	**CF11**	**CF10**	**CF9**	**CF8**	**CF7**	**CF6**	**CF5**	**CF4**	**CF3**	**CF2**	**CF1**	**CF0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CFn (n = 0-31)	n	rw	**Configuration flag x (x=0-2) for alarm n belonging to alarm group 2.** The configuration flags 0, 1 and 2 must be used together to define the behavior of the SMU when a fault state is reported by the alarm n belonging to this group (see **“Alarm Configuration” on Page 10-41**). *Note: **If the alarm is specified as reserved in “Alarm Signals” on Page 10-24, software shall configure the behavior to “No Action”.*** 0_B Configuration flag x (x=0-2) is set to 0 1_B Configuration flag x (x=0-2) is set to 1

SMU Alarm Configuration Registers related to Alarm Group 3.

SMU_AG3CFx (x=0-0)
Alarm Configuration Register (124_H + x*04_H) Reset Value: 001E 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CF31	CF30	CF29	CF28	CF27	CF26	CF25	CF24	CF23	CF22	CF21	CF20	CF19	CF18	CF17	CF16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CFn (n = 0-31)	n	rw	**Configuration flag x (x=0-0) for alarm n belonging to alarm group 3.** The configuration flags 0, 1 and 2 must be used together to define the behavior of the SMU when a fault state is reported by the alarm n belonging to this group (see **"Alarm Configuration" on Page 10-41**). *Note: **If the alarm is specified as reserved in "Alarm Signals" on Page 10-24, software shall configure the behavior to "No Action".*** 0_B Configuration flag x (x=0-2) is set to 0 1_B Configuration flag x (x=0-2) is set to 1

Note: The reset value specifies the NMI internal action for the watchdog timeout alarms. The reset value is the same regardless of the number of watchdogs present in the device.

SMU Alarm Configuration Registers related to Alarm Group 3.

SMU_AG3CFx (x=1-1)
Alarm Configuration Register (124_H + x*04_H) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CF31	**CF30**	**CF29**	**CF28**	**CF27**	**CF26**	**CF25**	**CF24**	**CF23**	**CF22**	**CF21**	**CF20**	**CF19**	**CF18**	**CF17**	**CF16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CF15	**CF14**	**CF13**	**CF12**	**CF11**	**CF10**	**CF9**	**CF8**	**CF7**	**CF6**	**CF5**	**CF4**	**CF3**	**CF2**	**CF1**	**CF0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CFn (n = 0-31)	n	rw	**Configuration flag x (x=1-1) for alarm n belonging to alarm group 3.** The configuration flags 0, 1 and 2 must be used together to define the behavior of the SMU when a fault state is reported by the alarm n belonging to this group (see **“Alarm Configuration” on Page 10-41**). *Note: **If the alarm is specified as reserved in “Alarm Signals” on Page 10-24, software shall configure the behavior to “No Action”.*** 0_B Configuration flag x (x=0-2) is set to 0 1_B Configuration flag x (x=0-2) is set to 1

Note: The reset value specifies the NMI internal action for the watchdog timeout alarms. The reset value is the same regardless of the number of watchdogs present in the device.

SMU Alarm Configuration Registers related to Alarm Group 3.

SMU_AG3CFx (x=2-2)
Alarm Configuration Register (124_H + x*04_H) Reset Value: 001E 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CF31	CF30	CF29	CF28	CF27	CF26	CF25	CF24	CF23	CF22	CF21	CF20	CF19	CF18	CF17	CF16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CFn (n = 0-31)	n	rw	**Configuration flag x (x=2) for alarm n belonging to alarm group 3.** The configuration flags 0, 1 and 2 must be used together to define the behavior of the SMU when a fault state is reported by the alarm n belonging to this group (see **"Alarm Configuration" on Page 10-41**). *Note: **If the alarm is specified as reserved in "Alarm Signals" on Page 10-24, software shall configure the behavior to "No Action".*** 0_B Configuration flag x (x=0-2) is set to 0 1_B Configuration flag x (x=0-2) is set to 1

Note: The reset value specifies the NMI internal action for the watchdog timeout alarms. The reset value is the same regardless of the number of watchdogs present in the device.

SMU Alarm Configuration Registers related to Alarm Group 4.

SMU_AG4CFx (x=0-2)
Alarm Configuration Register (130_H + x*04_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CF31	**CF30**	**CF29**	**CF28**	**CF27**	**CF26**	**CF25**	**CF24**	**CF23**	**CF22**	**CF21**	**CF20**	**CF19**	**CF18**	**CF17**	**CF16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CF15	**CF14**	**CF13**	**CF12**	**CF11**	**CF10**	**CF9**	**CF8**	**CF7**	**CF6**	**CF5**	**CF4**	**CF3**	**CF2**	**CF1**	**CF0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CFn (n = 0-31)	n	rw	**Configuration flag x (x=0-2) for alarm n belonging to alarm group 4.** The configuration flags 0, 1 and 2 must be used together to define the behavior of the SMU when a fault state is reported by the alarm n belonging to this group (see **“Alarm Configuration” on Page 10-41**). *Note:* ***If the alarm is specified as reserved in “Alarm Signals” on Page 10-24, software shall configure the behavior to “No Action”.*** 0_B Configuration flag x (x=0-2) is set to 0 1_B Configuration flag x (x=0-2) is set to 1

SMU Alarm Configuration Registers related to Alarm Group 5.

SMU_AG5CFx (x=0-2)
Alarm Configuration Register ($13C_H$ + x*04_H) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CF31	**CF30**	**CF29**	**CF28**	**CF27**	**CF26**	**CF25**	**CF24**	**CF23**	**CF22**	**CF21**	**CF20**	**CF19**	**CF18**	**CF17**	**CF16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CF15	**CF14**	**CF13**	**CF12**	**CF11**	**CF10**	**CF9**	**CF8**	**CF7**	**CF6**	**CF5**	**CF4**	**CF3**	**CF2**	**CF1**	**CF0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CFn (n = 0-31)	n	rw	**Configuration flag x (x=0-2) for alarm n belonging to alarm group 5.** The configuration flags 0, 1 and 2 must be used together to define the behavior of the SMU when a fault state is reported by the alarm n belonging to this group (see **"Alarm Configuration" on Page 10-41**). *Note: **If the alarm is specified as reserved in "Alarm Signals" on Page 10-24, software shall configure the behavior to "No Action".*** 0_B Configuration flag x (x=0-2) is set to 0 1_B Configuration flag x (x=0-2) is set to 1

SMU Alarm Configuration Registers related to Alarm Group 6.

SMU_AG6CFx (x=0-2)
Alarm Configuration Register (148_H + x*04_H) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CF31	CF30	CF29	CF28	CF27	CF26	CF25	CF24	CF23	CF22	CF21	CF20	CF19	CF18	CF17	CF16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CFn (n = 0-31)	n	rw	**Configuration flag x (x=0-2) for alarm n belonging to alarm group 6.** The configuration flags 0, 1 and 2 must be used together to define the behavior of the SMU when a fault state is reported by the alarm n belonging to this group (see **"Alarm Configuration" on Page 10-41**). *Note:* ***If the alarm is specified as reserved in "Alarm Signals" on Page 10-24, software shall configure the behavior to "No Action".*** 0_B Configuration flag x (x=0-2) is set to 0 1_B Configuration flag x (x=0-2) is set to 1

10.5.4 SMU Alarm Configuration Registers (Fault Signaling Protocol)

SMU Fault Signalling Protocol Alarm Configuration Registers related to Alarm Group 0.

SMU_AG0FSP
FSP Configuration Register (180_H) Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FE31	**FE30**	**FE29**	**FE28**	**FE27**	**FE26**	**FE25**	**FE24**	**FE23**	**FE22**	**FE21**	**FE20**	**FE19**	**FE18**	**FE17**	**FE16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FE15	**FE14**	**FE13**	**FE12**	**FE11**	**FE10**	**FE9**	**FE8**	**FE7**	**FE6**	**FE5**	**FE4**	**FE3**	**FE2**	**FE1**	**FE0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
FEn (n = 0-31)	n	rw	**Fault signalling configuration flag for alarm n belonging to alarm group 0.** 0_B FSP disabled for this alarm event 1_B FSP enabled for this alarm event

SMU Fault Signalling Protocol Alarm Configuration Registers related to Alarm Group 1.

SMU_AG1FSP

FSP Configuration Register (184_H) Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FE31	**FE30**	**FE29**	**FE28**	**FE27**	**FE26**	**FE25**	**FE24**	**FE23**	**FE22**	**FE21**	**FE20**	**FE19**	**FE18**	**FE17**	**FE16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FE15	**FE14**	**FE13**	**FE12**	**FE11**	**FE10**	**FE9**	**FE8**	**FE7**	**FE6**	**FE5**	**FE4**	**FE3**	**FE2**	**FE1**	**FE0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
FEn (n = 0-31)	n	rw	**Fault signalling configuration flag for alarm n belonging to alarm group 1.** 0_B FSP disabled for this alarm event 1_B FSP enabled for this alarm event

SMU Fault Signalling Protocol Alarm Configuration Registers related to Alarm Group 2.

SMU_AG2FSP

FSP Configuration Register (188_H) Reset Value: $2000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FE31	**FE30**	**FE29**	**FE28**	**FE27**	**FE26**	**FE25**	**FE24**	**FE23**	**FE22**	**FE21**	**FE20**	**FE19**	**FE18**	**FE17**	**FE16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FE15	**FE14**	**FE13**	**FE12**	**FE11**	**FE10**	**FE9**	**FE8**	**FE7**	**FE6**	**FE5**	**FE4**	**FE3**	**FE2**	**FE1**	**FE0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
FEn (n = 0-31)	n	rw	**Fault signalling configuration flag for alarm n belonging to alarm group 2.** 0_B FSP disabled for this alarm event 1_B FSP enabled for this alarm event

SMU Fault Signalling Protocol Alarm Configuration Registers related to Alarm Group 3.

SMU_AG3FSP

FSP Configuration Register (18C$_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FE31	**FE30**	**FE29**	**FE28**	**FE27**	**FE26**	**FE25**	**FE24**	**FE23**	**FE22**	**FE21**	**FE20**	**FE19**	**FE18**	**FE17**	**FE16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FE15	**FE14**	**FE13**	**FE12**	**FE11**	**FE10**	**FE9**	**FE8**	**FE7**	**FE6**	**FE5**	**FE4**	**FE3**	**FE2**	**FE1**	**FE0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
FEn (n = 0-31)	n	rw	**Fault signalling configuration flag for alarm n belonging to alarm group 3.** 0_B FSP disabled for this alarm event 1_B FSP enabled for this alarm event

SMU Fault Signalling Protocol Alarm Configuration Registers related to Alarm Group 4.

SMU_AG4FSP

FSP Configuration Register (190_H) Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FE31	**FE30**	**FE29**	**FE28**	**FE27**	**FE26**	**FE25**	**FE24**	**FE23**	**FE22**	**FE21**	**FE20**	**FE19**	**FE18**	**FE17**	**FE16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FE15	**FE14**	**FE13**	**FE12**	**FE11**	**FE10**	**FE9**	**FE8**	**FE7**	**FE6**	**FE5**	**FE4**	**FE3**	**FE2**	**FE1**	**FE0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
FEn (n = 0-31)	n	rw	**Fault signalling configuration flag for alarm n belonging to alarm group 4.** 0_B FSP disabled for this alarm event 1_B FSP enabled for this alarm event

SMU Fault Signalling Protocol Alarm Configuration Registers related to Alarm Group 5.

SMU_AG5FSP

FSP Configuration Register (194_H) Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FE31	**FE30**	**FE29**	**FE28**	**FE27**	**FE26**	**FE25**	**FE24**	**FE23**	**FE22**	**FE21**	**FE20**	**FE19**	**FE18**	**FE17**	**FE16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FE15	**FE14**	**FE13**	**FE12**	**FE11**	**FE10**	**FE9**	**FE8**	**FE7**	**FE6**	**FE5**	**FE4**	**FE3**	**FE2**	**FE1**	**FE0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
FEn (n = 0-31)	n	rw	**Fault signalling configuration flag for alarm n belonging to alarm group 5.** 0_B FSP disabled for this alarm event 1_B FSP enabled for this alarm event

SMU Fault Signalling Protocol Alarm Configuration Registers related to Alarm Group 6.

SMU_AG6FSP

FSP Configuration Register (198_H) Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FE31	**FE30**	**FE29**	**FE28**	**FE27**	**FE26**	**FE25**	**FE24**	**FE23**	**FE22**	**FE21**	**FE20**	**FE19**	**FE18**	**FE17**	**FE16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FE15	**FE14**	**FE13**	**FE12**	**FE11**	**FE10**	**FE9**	**FE8**	**FE7**	**FE6**	**FE5**	**FE4**	**FE3**	**FE2**	**FE1**	**FE0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
FEn (n = 0-31)	n	rw	**Fault signalling configuration flag for alarm n belonging to alarm group 6.** 0_B FSP disabled for this alarm event 1_B FSP enabled for this alarm event

10.5.5 SMU Alarm Status Registers

SMU Alarm status registers related to the different alarm groups.

SMU_AGx (x=0-6)
Alarm Status Register **($1C0_H$ + x*04_H)** **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SF31	**SF30**	**SF29**	**SF28**	**SF27**	**SF26**	**SF25**	**SF24**	**SF23**	**SF22**	**SF21**	**SF20**	**SF19**	**SF18**	**SF17**	**SF16**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SF15	**SF14**	**SF13**	**SF12**	**SF11**	**SF10**	**SF9**	**SF8**	**SF7**	**SF6**	**SF5**	**SF4**	**SF3**	**SF2**	**SF1**	**SF0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
SFn (n = 0-31)	n	rwh	**Status flag for alarm n belonging to alarm group x (x=0-6).** 0_B Status flag n does not report a fault condition 1_B Status flag n reports a fault condition

Refer to **“Alarm Status Registers” on Page 10-44** for the conditions to set and reset the status flag by software.

10.5.6 SMU Alarm Debug Registers

SMU Alarm debug registers related to the different alarm groups.

SMU_ADx (x=0-6)
Alarm Status Register **(200_H + x*04_H)** **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DF31	**DF30**	**DF29**	**DF28**	**DF27**	**DF26**	**DF25**	**DF24**	**DF23**	**DF22**	**DF21**	**DF20**	**DF19**	**DF18**	**DF17**	**DF16**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DF15	**DF14**	**DF13**	**DF12**	**DF11**	**DF10**	**DF9**	**DF8**	**DF7**	**DF6**	**DF5**	**DF4**	**DF3**	**DF2**	**DF1**	**DF0**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

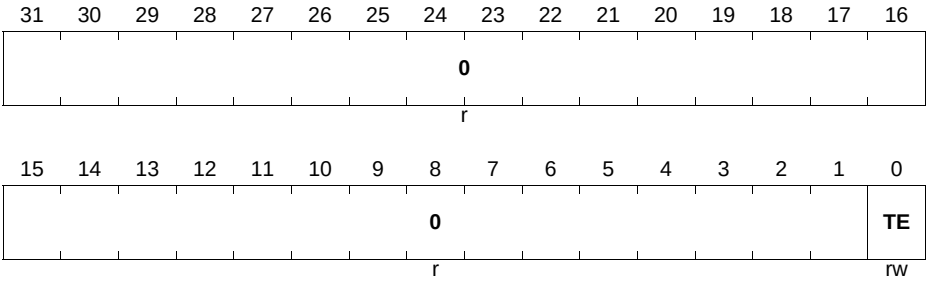
Field	Bits	Type	Description
DFn (n = 0-31)	n	rh	**Debug flag for alarm n belonging to alarm group x (x=0-6).** 0_B Status flag n does not report a fault condition 1_B Status flag n reports a fault condition

Note: Writing to this register has no effect

10.5.7 SMU Special Safety Registers: Register Monitor

SMU register controlling the test mode of the Safety Flip-Flop safety mechanism (also called Register Monitor).

SMU_RMCTL
Register Monitor Control **(300_H)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
TE	0	rw	**Test Enable.** This bit controls the timing of the test mode of the Safety Flip-Flop safety mechanism. This bit is broadcast to all modules that implement the Safety Flip-Flop safety mechanism. 0_B Test mode disabled 1_B Test mode enabled
0	[31:1]	r	**Reserved** Read as 0; should be written with 0

SMU register holding the test execution status of the Safety Flip-Flop safety mechanism.

SMU_RMEF
Register Monitor Error Flags (304_H) Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EF31	**EF30**	**EF29**	**EF28**	**EF27**	**EF26**	**EF25**	**EF24**	**EF23**	**EF22**	**EF21**	**EF20**	**EF19**	**EF18**	**EF17**	**EF16**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EF15	**EF14**	**EF13**	**EF12**	**EF11**	**EF10**	**EF9**	**EF8**	**EF7**	**EF6**	**EF5**	**EF4**	**EF3**	**EF2**	**EF1**	**EF0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
EFn (n = 0-31)	n	rwh	**Status flag related to the different instances of the Safety Flip-Flop safety mechanism.** In non-test mode (SMU_RMCTL.TE=0), the flag also reports a normal flip flop error condition. It can be used by software to identify the module where an error happened upon detection of an alarm event in ALM3[27]. The mapping of EFn flags to Modules is: EF0: SMU EF1: SCU EF2: MTU EF3...31 not used This flag can only be cleared by software, a set by software has no effect. 0_B Error flag n does not report a fault condition 1_B Error flag n reports a fault condition

SMU register holding the test status of the Safety Flip-Flop safety mechanism.

SMU_RMSTS

Register Monitor Self Test Status (308_H) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STS 31	**STS 30**	**STS 29**	**STS 28**	**STS 27**	**STS 26**	**STS 25**	**STS 24**	**STS 23**	**STS 22**	**STS 21**	**STS 20**	**STS 19**	**STS 18**	**STS 17**	**STS 16**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STS 15	**STS 14**	**STS 13**	**STS 12**	**STS 11**	**STS 10**	**STS 9**	**STS 8**	**STS 7**	**STS 6**	**STS 5**	**STS 4**	**STS 3**	**STS 2**	**STS 1**	**STS 0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
STSn (n = 0-31)	n	rwh	**Ready flag related to the different instances of the Safety Flip-Flop safety mechanism.** This bit indicates whether the Safety Flip-Flop test is finished or not (when SMU_RMCTL.TE=1). This bit can only be cleared by software, a set by software has no effect. The mapping of STSn flags to Modules is: STS0: SMU STS1: SCU STS2: MTU STS3...31 not used 0_B Self-test has not completed 1_B Self-test has completed

11 Program Memory Unit (PMU)

The Program Memory Unit (PMU) controls the Flash memory and the BootROM and connects them to the SRI crossbar.

The devices of the AURIX family have at least one PMU. This is named “PMU0”. Depending on the amount of Flash memory more PMUs are added which are named “PMU1”, and so on. All PMUs are independent from each other.

TC21x/TC22x/TC23x has 1 PMU(s). Throughout this document a generic PMU is specified. The **Chapter 11.2** lists the configuration parameters of each PMU and their values in the TC21x/TC22x/TC23x. When PMU functionality depends on a parameter this is indicated in the text.

This chapter has the following structure:

- Generic feature list and block diagram (**Chapter 11.1**).
- PMU configuration of the TC21x/TC22x/TC23x (**Chapter 11.2**).
- Functionality of the BootROM (**Chapter 11.3**).
- Functionality of the tuning protection (**Chapter 11.4**).
- Functionality of the Flash memory (**Chapter 11.5**).
- Signaling to the Safety Management Unit “SMU” (**Chapter 11.6**).
- Register Set (**Chapter 11.7**).
- Application Hints (**Chapter 11.8**).

11.1 Generic Feature List

In general a PMU has the following features:

- Only in PMU0: BootROM.
- Program Flash “PFlash” with one or more banks “PFx”.
- Data Flash “DFlash” with one or more banks “DFx” containing:
  - Sectors for EEPROM emulation (only in PMU0).
  - UCB sectors for protection data.
- One Flash Standard Interface “FSI” that controls all Flash operations.
- Access control for safety and security for all assigned memories.
- Only in PMU0: Tuning protection (commonly called “Secure Watchdog”).
- Configuration and control registers.
- SRI interface(s) to all included resources. Separate SRI interfaces for DFlash and for each bank of PFlash.

Figure 11-1 shows an abstracted block diagram of PMU, FSI, Flash and BootROM.

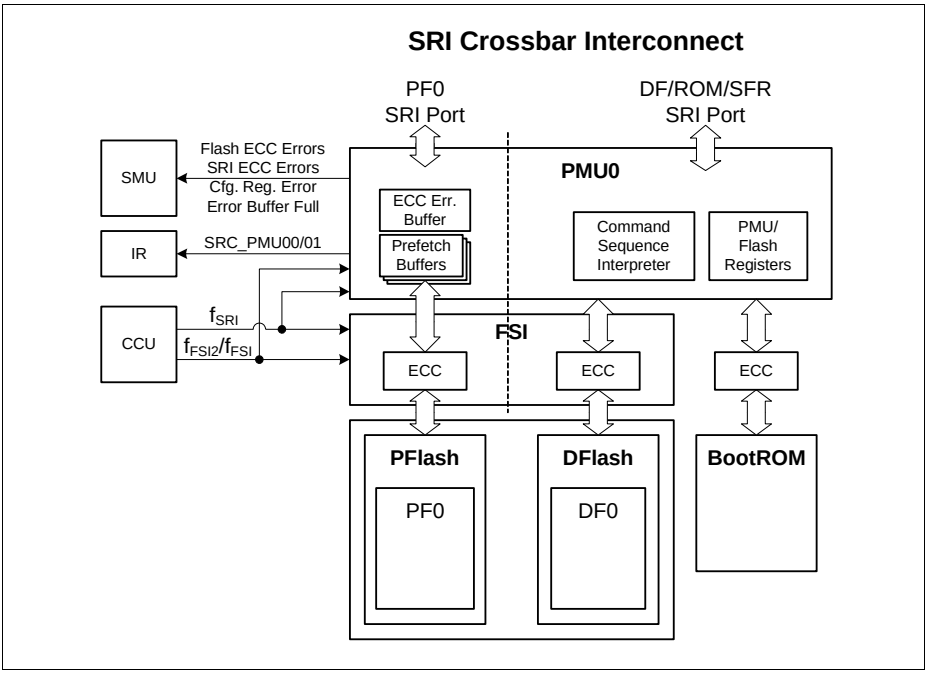


Figure 11-1 PMU/FSI/Flash Basic Block Diagram

11.2 PMU Configuration of TC21x/TC22x/TC23x

This chapter describes the superset of a range of devices with the following configuration:

- Number of PMUs: 1

Configuration of PMU0:

- PF0: 2 MByte.
- DF0 consisting of:
  - DF_EEPROM: 128 KByte (16 logical sectors “EEPROM0 … EEPROM15”).
  - DF_UCB: Flash area for protection data (16 logical sectors “UCB0 … UCB15” with 1 KByte each).

Address Map

The following table(s) contains the address map.

Table 11-1 Address Map of PMU0

Start Address	Size	Range	Start Address Symbol
$8000\ 0000_H$	2 MByte	PF0	AC_PF0
$8FFF\ 8000_H$	32 KByte	BootROM	AC_BROM0
$A000\ 0000_H$	2 MByte	PF0	AN_PF0
$AF00\ 0000_H$	128 KByte	DF0 (DF_EEPROM)[1)]	AN_DFlash_B0
$AF10\ 0000_H$	16 KByte	DF0 (DF_UCB)	
$AF11\ 0000_H$	64 KByte	Not implemented (Reserved for DF1)	AN_DFlash_B1
$AFFF\ 8000_H$	32 KByte	BootROM	AN_BROM0
$F800\ 0500_H$	256 Byte	PMU Registers	
$F800\ 1000_H$	5 KByte	Flash Registers	
$FF11\ 0000_H$	64 KByte	Not implemented (Reserved for DF1, HSM private access)	AN_DFlash_B1F

1) In derivates with reduced amount of DF0 (see data sheet) the available range is contiguous and starts also AN_DFlash_B0.

Derived Devices

This specification describes a superset of a range of devices. The concrete number and size of implemented Flash ranges is given in the respective Data Sheet.

In devices with PFlash banks of reduced size the sectors (see **Chapter 11.5.2.1**) with higher sector number are unavailable. The complete PFlash range is contiguous in all devices starting on AC_PF0/AN_PF0.

The “unavailable” PFlash sectors cannot exist or can be blocked during device manufacturing (the PMU handles these sectors then as not existing) or they stay accessible. In the latter case these sectors are delivered in the erased state. Full reliability is not ensured. These sectors shall not be read, programmed or erased. The write and OTP protection can be activated for these sectors to prevent later modification.

In devices with DFlash of reduced size the sectors EEPROMx (see **Chapter 11.5.2.2**) with higher sector number x are unavailable (the whole DF_EEPROM range stays contiguous).

The “unavailable” DFlash sectors cannot exist or can be blocked during device manufacturing (the PMU handles these sectors then as not existing) or they stay accessible. In the latter case these sectors are delivered in the erased state.

The available DFlash sectors must be used round-robin to prevent accumulation of disturbs in unevenly cycled sectors (see **Chapter 11.5.2.2**).

11.2.1 Features of the BootROM

The BootROM contains the firmware:

- Startup software “SSW”: this software is executed at every reset.
- Test firmware: factory test routines for Infineon purposes.

Additionally the BootROM and is control logic implement the Tuning protection feature (“Secure Watchdog”).

11.2.2 Features of the Program and Data Flash

Depending on the PMU configuration the following Flash features are implemented. Timing and reliability figures are just indicative. Binding data can be found in the “Data Sheet”.

11.2.2.1 Program Flash Features:

Summary of Program Flash features:

- Consists of 1 bank(s). All banks are concurrently readable.
- Separate SRI ports, ECC decoders and pre-fetch logic for each bank.
- Commonly used for instructions and constant data.
- High throughput burst read based on a 256-bit Flash access (see parameter table t_{PF}).
- Application optimized sector structure with logical sectors ranging from 16 KBytes to 256 KBytes.
- Write protection separately configurable per logical sector (one-time programmable and 256-bit password protected).
- Password based read protection combined with write protection.
- Sectors (one 16 KB and two 64 KB) configurable as HSM code sectors by setting HSM_exclusive flag.
- Separate read, write and OTP protection for the HSM code sectors.
- Erase counter counts each sector erase (saturating).
- Fast programming of 32 byte pages (see Data Sheet t_{PRP}).
- High throughput burst programming of 256 byte units (see Data Sheet t_{PRPB}).
- Erase time per logical sector: see Data Sheet t_{ERP}.
- For development purposes sectors may incur more program/erase cycles than for productive purposes. Additionally further restrictions apply as an increased failure rate and lower retention.
- High throughput erase by multi-sector erase commands: see Data Sheet t_{MERP}.
- Erase and program performed by a Flash specific control logic independent of the CPU.

- Fast suspend erase to read and suspend burst programming to read command.
- Erase verify command.
- End of erase and program operations reported by interrupt.
- Configurable stall or bus error generation when reading busy PFlash banks.
- Dynamic correction of single-bit and double-bit errors and detection of triple-bit errors (“DEC-TED”).
- “Safe read path” ensuring detection of transient and permanent errors including addressing faults.
- ASIL-D optimized ECC for detection of >99% of faults in the white-noise model.
- Error reporting to the SMU, additionally local status flags and bus error generation.
- Margin reads for quality assurance.
- Delivery in the erased state.
- Configurable wait-state configuration.
- Endurance and retention figures are documented in the Data Sheet.
- Pad supply voltage used for program and erase. Separate programming modes for 5V and 3.3V supplied devices.

11.2.2.2 Data Flash Features

Summary of Data Flash features:

- Contains 16 EEPROMx sectors commonly used for EEPROM emulation (data storage at application run-time).
- UCBx sectors used for protection installation and the erase-cycle counter.
- Read-only UCB configured by Infineon with unique chip identifier and trimming data.
- Password based read protection combined with write protection.
- Flash read access based on 64-bit reads (see Data Sheet t_{DF}).
- Read path separated from PFlash (DFlash reads don’t influence PFlash read accesses).
- Burst read is not supported.
- Requested read (data reading performed in background, read data supplied in registers).
- Fast programming of 8 byte pages (see Data Sheet t_{PRD}).
- High throughput burst programming of 32 byte units (see Data Sheet t_{PRDB}).
- Erase time per sector: see Data Sheet t_{ERD}.
- High throughput erase by multi-sector erase commands: see Data Sheet t_{MERD}.
- Erase and program performed by a Flash specific control logic independent of the CPU.
- Fast suspend erase to read command.
- End of erase and program operations reported by interrupt.
- Dynamic correction of single-bit, double-bit and triple-bit errors and detection of quad-bit errors (“TEC-QED”).
- Error reporting to the SMU, additionally local status flags and bus error generation.
- Margin reads for quality assurance.

- Delivery in the erased state.
- Configurable wait-state configuration.
- The high endurance (see Data Sheet N_E) is granted under the condition of a robust EEPROM emulation algorithm (see corresponding section in the PMU chapter of the User's Manual).
- Endurance and retention figures are documented in the Data Sheet.
- Pad supply voltage used for program and erase.

11.3 BootROM

The content of the BROM is described in a separate chapter.

The BROM is readable and executable for user software but its functions shall only be executed when especially advised by Infineon.

All write accesses to the BootROM are refused with a bus error.

Its content is protected with ECC having SEC-DED capability and address error detection.

In the SRI address range it is mapped to the following start addresses:

- AC_BROM0 (cached address range).
- AN_BROM0 (non-cached address range).

11.4 Tuning Protection

The special tuning protection support represents a security function provided additionally to Flash read/write/OTP protection.

For details on the tuning protection please contact your Infineon representative.

11.5 Flash

This chapter introduces the Flash memory of the TC21x/TC22x/TC23x. It is split into the following sections:

- Definition of terms (**Chapter 11.5.1**): what means “program”, “erase”, “sector”, “pages”, …
- Structure of a Flash module (**Chapter 11.5.2**): separation into “banks”, “sectors”, “word-lines”, “pages”, …
- Reading Flash (**Chapter 11.5.3**).
- Command sequences for Flash (**Chapter 11.5.4**): programming, erasing, handling protection.
- Flash protection (**Chapter 11.5.5**): read and write protection.
- Data integrity and safety (**Chapter 11.5.6**): ECC and margin checks.
- Interrupt and traps (**Chapter 11.5.7**).
- Reset and startup (**Chapter 11.5.8**).
- Power reduction by sleep and idle (**Chapter 11.5.9**).

11.5.1 Definition of Terms

The description of Flash memories uses a specific terminology for operations and the hierarchical structure.

Flash Operation Terms

- **Erasing**: The erased state of a Flash cell is logical ‘0’. Forcing a cell to this state is called “erasing”. Depending on the Flash area always complete physical sectors, logical sectors or word-lines are erased. All Flash cells in this area incur one “cycle” that counts for the “endurance”.
- **Programming**: The programmed state of a cell is logical ‘1’. Changing an erased Flash cell to this state is called “programming”. The 1-bits of a page are programmed concurrently.
- **Retention**: This is the time during which the data of a flash cell can be read reliably. The retention time is a statistical figure that depends on the operating conditions of the device (e.g. temperature profile) and is affected by operations on other Flash cells in the same word-line and physical sector. With an increasing number of program/erase cycles (see endurance) the retention is lowered.
- **Endurance**: As described above the data retention is reduced with an increasing number of program/erase cycles. The maximum number of program/erase cycles of each Flash cell is called “endurance”. As said for the retention it is a statistical figure that depends on operating conditions and the use of the flash cells and not to forget on the required quality level. The endurance is documented in the data sheet separately for PFlash logical sectors, UCBs and EEPROM sectors.

Flash Structure Terms

- **Flash Module**: A PMU contains one “Flash module” with its own operation control logic.
- **Bank**: A “Flash module” contains separate “banks”. In the PFlash there are one or more PFx banks and in the DFlash two DFx banks. “Banks” support concurrent operations with some limitations due to common logic.
- **Sub-Sector**: A Flash “bank” consists of sub-sectors. There are “physical sub-sectors” which are isolated from each other and “logical sub-sectors”. A PFlash bank is built from 512 KByte physical sub-sectors. A DFlash bank is separated into logical sub-sectors.
- **Logical Sector**: The sub-sectors are further separated into “logical sectors” which are groups of word-lines. They can be erased with a single operation.
- **Sector**: The plain term “sector” means “logical sector”.
- **User Configuration Block “UCB”**: A “UCB” is a specific logical sector. It is part of a DF_UCB sub-sector of bank DF0. It contains the protection settings and other data configured by the user.
- **Configuration Sector**: The “configuration sector” is a logical sector of the PFlash. It is not directly accessible to the user.
- **Page**: In the PFlash a page is an aligned group of 32 bytes and in DFlash of 8 bytes. It is the smallest unit that can be programmed (ECC width).
- Program **Burst**: A “burst” is the maximum amount of data that can be programmed with one command. The programming throughput is higher than for programming single pages. A burst in PFlash consists of 8 pages (256 bytes) and in DFlash 4 pages (32 bytes).
- **Word-Line**: In the PFlash and DFlash a word-line is an aligned group of 512 bytes.

11.5.2 Flash Structure

The PMU0 of TC21x/TC22x/TC23x contains the following Flash banks. The offset address of each sector is relative to the base address of its bank which is given in **Table 11-1**. In devices of the same family the Flash banks keep the same base address.

11.5.2.1 PFlash

All banks PFx of the PFlash of all PMUs are based on the same sectorization (see **Table 11-2**). PFx banks with reduced capacity are built by cutting off sectors at the end. A 2 MByte bank implements logical sectors S0 to S26, a 1.5 MByte bank logical sectors S0 to S24 and a 1 MByte bank logical sectors S0 to S22. The implemented bank sizes are described in the Data Sheet.

Table 11-2 Sector Structure of PFx of PMU0

Logical Sector	Phys. Sub-Sector	Size	Offset Address[1)]
S0	PS0 (512 KB)	16 KB	$00'0000_H$
S1		16 KB	$00'4000_H$
S2		16 KB	$00'8000_H$
S3		16 KB	$00'C000_H$
S4		16 KB	$01'0000_H$
S5		16 KB	$01'4000_H$
S6		16 KB	$01'8000_H$
S7		16 KB	$01'C000_H$
S8		32 KB	$02'0000_H$
S9		32 KB	$02'8000_H$
S10		32 KB	$03'0000_H$
S11		32 KB	$03'8000_H$
S12		32 KB	$04'0000_H$
S13		32 KB	$04'8000_H$
S14		32 KB	$05'0000_H$
S15		32 KB	$05'8000_H$
S16		64 KB	$06'0000_H$
S17		64 KB	$07'0000_H$
S18	PS1 (512 KB)	64 KB	$08'0000_H$
S19		64 KB	$09'0000_H$
S20		128 KB	$0A'0000_H$
S21		128 KB	$0C'0000_H$
S22		128 KB	$0E'0000_H$
S23	PS2 (512 KB)	256 KB	$10'0000_H$
S24		256 KB	$14'0000_H$
S25	PS3 (512 KB)	256 KB	$18'0000_H$
S26		256 KB	$1C'0000_H$

1) Offset with respect to AC_PFx and AN_PFx (see **Table 11-1**).

The sectors S0, S3, S7 and S8 of PF0 in PMU0 contain the BMI headers.

The sectors S6, S16 and S17 of PF0 in PMU0 are used as HSM code sectors in devices with HSM. Their access control (read, program, erase) is different from other PFlash sectors.

The sector S5 of PF0 in PMU0 has a specific purpose with an enabled Tuning Protection.

11.5.2.2 DFlash of PMU0

Bank DF0 of the DFlash of PMU0 (see **Table 11-3**).

Attention: In the DF_EEPROM all sectors must be used round-robin in order to prevent the accumulation of erase disturbs in static sectors[1]. In derivates with reduced amount of DFlash all sectors in this reduced DFlash range must be used round-robin.

Attention: In DF_EEPROM the overall number of erase commands over lifetime is limited by the parameter N_{ERD0}[2], in DF_HSM by N_{ERD1}[3].

Attention: In DF_EEPROM and DF_HSM at most 384 KByte shall be erased with one erase command.

Table 11-3 Sector Structure of DFlash Bank 0

Logical Sector	Log. Sub-Sector	Size	Offset Address[1]
EEPROM0	DF_EEPROM	8 KByte	$00'0000_H$
EEPROM1		8 KByte	$00'2000_H$
EEPROM2		8 KByte	$00'4000_H$
EEPROM3		8 KByte	$00'6000_H$
…		…	…
EEPROM15		8 KByte	$01'E000_H$

1) The parameter N_{DFD} (see Electrical Parameters or Data Sheet) documents the erase disturb limit. It is chosen higher than needed for pure round-robin usage to cover also error cases (e.g. repetition of erase commands on the same range due to power failures during operation).

2) This parameter is chosen to not reduce the endurance when always with one erase command at least 1/6th of the available DF_EEPROM is erased.

3) This parameter is chosen to not reduce the endurance when always with one erase command at least 1/4th of the available DF_HSM is erased

Table 11-3 Sector Structure of DFlash Bank 0 (cont'd)

Logical Sector	Log. Sub-Sector	Size	Offset Address[1)]
UCB0 = UCB_PFlash	DF_UCB	1 KByte	$10'0000_H$
UCB1 = UCB_DFlash		1 KByte	$10'0400_H$
UCB2 = UCB_HSMCOTP		1 KByte	$10'0800_H$
UCB3 = UCB_OTP		1 KByte	$10'0C00_H$
UCB4 = UCB_IFX		1 KByte	$10'1000_H$
UCB5 = UCB_DBG		1 KByte	$10'1400_H$
UCB6 = UCB_HSM		1 KByte	$10'1800_H$
UCB7 = UCB_HSMCFG		1 KByte	$10'1C00_H$
UCB8		1 KByte	$10'2000_H$
UCB9		1 KByte	$10'2400_H$
UCB10		1 KByte	$10'2800_H$
UCB11		1 KByte	$10'2C00_H$
UCB12		1 KByte	$10'3000_H$
UCB13		1 KByte	$10'3400_H$
UCB14		1 KByte	$10'3800_H$
UCB15		1 KByte	$10'3C00_H$

1) Offset with respect to AN_DFlash_B0 (see **Table 11-1**).

11.5.3 Flash Read Access

Flash banks that are active and in read mode can be directly (memory mapped) read as ROM.

The wait cycles for the Flash access must be configured as described in (**Chapter 11.5.3.3**).

The PFlash allows SRI bursts BTR4 and BTR2 and single transfers.

The Tricore uses BTR4 for code fetches from the cached address range in order to fill one cache line. Code fetches from the non-cached area are also performed with BTR4. Data reads from the cached range are performed with BTR4. Data reads from the non-cached address range are performed with single transfers.

The DFlash allows only single transfers. Therefore they are only accessible in the non-cached address range and Tricore can't fetch instructions from them.

Read accesses from Flash can be blocked by the read protection (see **Chapter 11.5.5**).

Read accesses from busy Flash banks are not possible (see **Chapter 11.5.4.1**).

ECC errors can be detected and corrected (see **Chapter 11.5.6**).

The DFlash sectors EEPROMx can be also read with requested reads (see **Chapter 11.5.3.4**).

11.5.3.1 Read Ports

For optimum performance the PFlash is read via dedicated SRI slave ports. Each PFp bank is assigned to one SRI port determined by its address (see **Table 11-4**). The DFlash, the BootROM and the registers share a separate SRI slave port.

Table 11-4 Memory Range to SRI Port Assignment

SRI Port	Address Range	Memory Ranges
p = 0 mapped to SCI 7	8000 0000$_H$ – 801F FFFF$_H$	PF0 (cached)
p = 0 mapped to SCI 7	A000 0000$_H$ – A01F FFFF$_H$	PF0 (non-cached)
mapped to SCI 6	8FFF 8000$_H$ – 8FFF FFFF$_H$ AF00 0000$_H$ – AF01 FFFF$_H$ AF10 0000$_H$ – AF10 3FFF$_H$ AF11 0000$_H$ – AF11 FFFF$_H$ AFFF 8000$_H$ – AFFF FFFF$_H$ F800 0500$_H$ – F800 05FF$_H$ F800 1000$_H$ – F800 23FF$_H$ FF11 0000$_H$ – FF11 FFFF$_H$	BootROM DF0 (DF_EEPROM) DF0 (DF_UCB) reserved for DF1 BootROM PMU Registers Flash Registers reserved for HSM private Flash command interface and DF1.

Each SRI port can be independently read. Each has its own path to its Flash range and its own ECC decoding logic.

As additional performance increasing measure each of the PFp ports has its own prefetching logic and a set of 3 read buffers.

Every read buffer is assigned by configuration (see **Chapter 11.7.2.7**) to one bus master. Additionally all read data passes through a pipeline stage in the FSI called “global read buffer”. The read buffers can be filled with a single PFlash read which delivers 256 data bits.

Upon finishing a pre-fetch and with no request from the bus the PMU reads the sequentially following 256 bits into the global read buffer. Buffer hits in this read buffer are also supported and don’t cause a Flash read. When this read buffer is also filled the PMU sends already the address for the following 256 bits to the Flash and keep this address stable (this feature can be disabled with FCON.IDLE). A hit with this address is also detected and will reduce the Flash access time.

All read buffers are invalidated upon executing a command sequence that changed PFlash content. The read buffers are not invalidated by changes to the configuration registers (e.g. changing wait-states or changing the ECC correction).

11.5.3.2 DFlash, BootROM Read Port

The other resources (DFlash, BootROM, registers, etc) share a separate SRI slave port independent of the PFlash read ports.

Read accesses to the DFlash are always serviced by reading the Flash. Buffer hits are not supported.

11.5.3.3 Configuring Flash Wait Cycles

The Flash read path including the ECC decoders and read buffer accesses use the following clocks:

- PFx read ports: clocked with f_{FSI2}.
- DFlash, BootROM, register read port: clocked with f_{FSI}.

The wait cycles in register FCON are counted with this reference clock. The minimum number of wait cycles can be calculated based on the delays given in the Data Sheet or Electrical Specification as described in **Table 11-5**.

Note: Please note that in case of using the FM-PLL the maximum frequency of these clocks has to enter the calculation.

Attention: After System Reset and Power-On Reset the wait cycles are configured for the clock configuration used during startup, i.e. f_{FSI} and f_{FSI2} running with maximum 100 MHz. Before changing to higher clock frequencies this default configuration must be changed.

The clock f_{FSI2} can be configured to be identical to f_{SRI} or f_{FSI}. Selecting the higher clock frequency increases the read performance.

Table 11-5 Wait Cycle Calculation

	Register Field	Minimum Value[1)]
PFlash read access delay	FCON.WSPFLASH	$Ceil(t_{PF} * f_{FSI2}) - 1$
PFlash ECC decode delay	FCON.WSECPF	$Ceil(t_{PFECC} * f_{FSI2}) - 1$
DFlash read access delay	FCON.WSDFLASH	$Ceil(t_{DF} * f_{FSI}) - 1$
DFlash ECC decode delay	FCON.WSECDF	$Ceil(t_{DFECC} * f_{FSI}) - 1$

1) The Ceil(r) function rounds a real number up, i.e. the result is the smallest integer not less than the real argument. The mathematical function name is “Ceiling(r)”.

Examples

Example configuration for t_{PF} = 30 ns, t_{PFECC} = 10 ns, t_{DF} = 100 ns, t_{DFECC} = 20 ns with f_{FSI2} = 200 MHz and f_{FSI} = 100 MHz:

- FCON.WSPFLASH = 5
- FCON.WSECPF = 1
- FCON.WSDFLASH = 9
- FCON.WSECDF = 1

11.5.3.4 Requested DFlash Read

The direct memory mapped DFlash read stalls the requesting master for the duration of the read access. Applications that favor latency over throughput can use the “requested read” feature. This supports interrupt driven background reads from the DFlash.

In devices with DF_HSM a separate set of registers exist to perform requested reads on the HSMx sectors (**Chapter 11.7.2.11**).

The start address is written into RRAD.ADD. RRCT.CNT contains the number of 8 byte aligned transfers that will be performed.

The reading of each 8 byte block is started by setting RRCT.STRT. When the PMU starts the read process is clears RRCT.STRT and sets RRCT.BUSY.

When the data is available in RRD0-1 the flag RRCT.BUSY is cleared and RRCT.DONE is set. If RRCT.EOBM is set the requested read interrupt (SRC_PMU1) is triggered. The address in RRAD.ADD is incremented to the next address and the counter RRCT.CNT is decremented by 1.

The reading of the next 8 byte block has to be triggered again by setting RRCT.STRT. This clears RRCT.DONE.

The requested read can be aborted by setting RRCT.STP. This clears RRCT.STRT, RRCT.BUSY and RRCT.DONE. The requested read interrupt is not triggered.

An error is reported by RRCT.ERR when the read access addresses an illegal address (outside of the assigned DFlash ranges, insufficient access rights) or when started with CNT=0. This sets also RRCT.DONE.

The occurrence of an uncorrectable ECC error sets RRCT.ERR. The ECC error flags in the FSR are not influenced by requested reads.

Direct reads to the DFlash are not blocked by ongoing requested reads. They are executed with higher priority than requested reads.

11.5.4 Flash Operations

The Flash memory knows operations for reading, changing data (program/erase) and handling protection settings. This section and the following describe only the features as such. How to use them is described in the application notes (**Chapter 11.8**).

11.5.4.1 Modes of Operation

A Flash module can be in one of the following states:

- Active (normal) mode.
- Sleep mode (see **Chapter 11.5.8**).

In sleep mode write and read accesses to all Flash ranges of this PMU are refused with a bus error.

When the Flash module is in normal mode a Flash bank can be in one of these modes:

- Read mode.
- Command mode.

In read mode a Flash bank can be read and command sequences are interpreted. In read mode a Flash bank can additionally enter page mode which enables it to receive data for programming.

In command mode an operation is performed. During its execution the Flash bank reports BUSY in FSR. In this mode read accesses to this Flash bank are refused with a bus error or the ready is suppressed until BUSY clears. This can be configured (see **FCON**.STALL). At the end of an operation the Flash bank returns to read mode and BUSY is cleared. Only operations with a significant duration (shown in the command documentation) set BUSY.

Note: Using FCON.STALL = "STALL" is not recommended because the stalled CPU is inoperable and also the PMU SRI port is blocked as long the ready is suppressed. This feature should be only used under controlled conditions by a Flash loader.

*Note: Using FCON.STALL = "STALL" together with Flash sleep (**Chapter 11.5.9**) is prohibited as it might lead to device hang-up.*

In command mode further command sequences in this flash module are not allowed. They are refused with a bus error.

Flash modules of different PMUs are completely independent.

Register read and write accesses are not affected by these modes.

11.5.4.2 Command Sequences

All Flash operations except memory mapped reads are performed with command sequences. Every write access to the DF0 memory range is interpreted as command cycle belonging to a command sequence.

Write accesses to the PFlash memory range are refused with bus error.

Command sequences consist of 1 to 9 command cycles. The command interpreter checks that a command cycle is correct in the current state of command interpretation. Else a SQER is reported.

When the command sequence is accepted the last command cycle finishes read mode and the Flash bank transitions into command mode.

Command sequences can be blocked by the register access protection (see **Chapter 11.5.5.2**).

Generally when the command interpreter detects an error it reports a sequence error by setting FSR.SQER. Then the command interpreter is reset and a page mode is left. The next command cycle must be the 1st cycle of a command sequence. The only exception is “Enter Page Mode” when a Flash is already in page mode (see below).

11.5.4.3 HSM Command Interface

In devices with DF_HSM there is a separate command interface (interpreting writes to the DF1 memory range) to perform operations on this Flash bank. In devices without DF_HSM this command interface doesn’t accept any commands.

11.5.4.4 Command Sequence Definitions

The command sequence descriptions use the following nomenclature (symbolic assembly language):

ST addr, data: Symbolic representation of a command cycle moving “data” to “addr”.

The parameter “addr” is defined as followed:

- **$CCCC_H$**: The “addr” must point into the DFlash. The last 16 address bits must match $CCCC_H$.

The parameter “data” can be one of the following:

- **PA**: Absolute start address of the Flash page. Must be aligned to burst size for “Write Burst” or to the page size for “Write Page”.
- **SA**: Absolute start address of a Flash sector. Allowed are the PFlash sectors Sx and the DFlash sectors EEPROMx, UCBx.
- **WD**: 64-bit or 32-bit write data to be loaded into the page assembly buffer.
- **xxYY**: 8-bit write data as part of a command cycle. Only the byte “YY” is used for command interpretation. The higher order bytes “xx” are ignored.
  - **xx5y**: Specific case for “YY”. The “y” can be “0_H” for selecting the PFlash or “D_H” to select the DFlash.
  - **xxnn**: Specific case for “YY”. The “nn” quantifies the number of logical sectors that are erased or verified. This number underlies certain restrictions (see **Erase Logical Sector Range** on **Page 11-20**).
- **UC**: Identification of the UCBx for which the password checking shall be performed: $xx00_H$ for UCB0 = UCB_PFlash, $xx01_H$ for UCB1 = UCB_DFlash, $xx05_H$ for UCB5 = UCB_DBG.
- **PWx**: 32-bit word of a 256-bit password.

When using for command cycles 64-bit transfers the “data” is expected in the correct 32-bit word as indicated by the address “addr”.

Command Sequence Overview Table

The **Table 11-6** summarizes all commands sequences. The following sections describe each command sequence in detail.

Table 11-6 Command Sequences for Flash Control

Command Sequence		1. Cycle	2./6. Cycle	3./7. Cycle	4./8. Cycle	5./9. Cycle	6. Cycle
Reset to Read	Address Data	.5554 ..xx**F0**					
Enter Page Mode	Address Data	.5554 ..xx**5y**					
Load Page	Address Data	.55F0 1) **WD**					
Write Page	Address Data	.AA50 **PA**	.AA58 ..xx**00**	.AAA8 ..xx**A0**	.AAA8 ..xx**AA**		
Write Page Once	Address Data	.AA50 **PA**	.AA58 ..xx**00**	.AAA8 ..xx**A0**	.AAA8 ..xx**9A**		
Write Burst	Address Data	.AA50 **PA**	.AA58 ..xx**00**	.AAA8 ..xx**A0**	.AAA8 ..xx**7A**		
Erase Logical Sector Range	Address Data	.AA50 **SA**	.AA58 ..xx**nn**	.AAA8 ..xx**80**	.AAA8 ..xx**50**		
Verify Erased Logical Sector Range	Address Data	.AA50 **SA**	.AA58 ..xx**nn**	.AAA8 ..xx**80**	.AAA8 ..xx**5F**		
Resume Prog/Erase	Address Data	.AA50 **PA/SA**	.AA58 ..xx**nn**	.AAA8 ..xx**70**	.AAA8 ..xx**CC**		
Disable Protection	Address Data	.553C **UC**	.553C **PW0/4**	.553C **PW1/5**	.553C **PW2/6**	.553C **PW3/7**	
Resume Protection	Address Data	.5554 ..xx**F5**					
Clear Status	Address Data	.5554 ..xx**FA**					

1) Address depends on data width of WD (see description of Load Page below).

Reset to Read

Calling:

- ST 5554_H, $xxF0_H$

Function:

This function resets the addressed command interpreter to its initial state (i.e. the next command cycle must be the 1st cycle of a sequence). A page mode is aborted.

This command is the only that is accepted without SQER when the command interpreter has already received command cycles of a different sequence but is still not in command mode. Thus “Reset to Read” can cancel every command sequence before its last command cycle has been received.

The error flags of **FSR** (OPER, SQER, PROER, PFSBER, PFDBER, PFMBER, DFSBER, DFDBER, DFTBER, DFMBER, ORIER, PVER, EVER) are cleared. The flags can be also cleared in the status registers without command sequence.

Enter Page Mode

Calling:

- ST 5554_H, $xx5y_H$

Function:

The PFlash or the DFlash assembly buffer enter page mode selected by the parameter “y”.

The write pointer of the page assembly buffer is set to 0, its previous content is maintained.

The page mode is signaled by the flags PFPAGE/DFPAGE in the FSR separately for PFlash and DFlash.

If a new “Enter Page Mode” command sequence is received while any Flash is already in page mode SQER is set, the existing page mode is aborted, and the new “Enter Page Mode” sequence is correctly executed (i.e. in this case the command interpreter is not reset).

Load Page

Calling:

- ST $55F0_H$/$55F4_H$, WD

Function:

Loads the data “WD” into the page assembly buffer and increments the write pointer to the next position.

All WD transfers for one page must have the same width (either all 32-bit or all 64-bit). Else the transfer is refused with SQER.

When using 64-bit transfers then all have to address the offset address $55F0_H$.

When using 32-bit transfers then the lower order word has to be written first to offset $55F0_H$ and then the higher order word to offset $55F4_H$. This alternating pattern is repeated for all WD.

If “Load Page” is called more often than allowed by the available buffer space of 256 bytes the overflow data is discarded, the page mode is not left. This overflow is reported by the following Write Page/Burst command with SQER.

Write Page

Calling:

- ST $AA50_H$, PA
- ST $AA58_H$, $xx00_H$
- ST $AAA8_H$, $xxA0_H$
- ST $AAA8_H$, $xxAA_H$

Function:

This function starts the programming process for one page with the data transferred previously by “Load Page” commands. Upon entering command mode the page mode is finished (indicated by clearing the corresponding PAGE flag) and the BUSY flag of the bank is set.

A page shall be programmed only once after erasing.

This command is refused with SQER when the addressed Flash is not in page mode.

SQER is also issued when PA addresses an unavailable Flash range or when PA does not point to a legal page start address.

If after “Enter Page Mode” too few data, no data or too much data was transferred to the assembly buffer with “Load Page” then “Write Page” programs the page but sets SQER. Missing data is programmed with the previous content of the assembly buffer.

When the page “PA” is located in a sector with active write protection or the Flash module has an active global read protection the execution fails and PROER is set.

When the programming process incurs an error the flag PVER is set.

Write Page Once

Calling:

- ST $AA50_H$, PA
- ST $AA58_H$, $xx00_H$
- ST $AAA8_H$, $xxA0_H$
- ST $AAA8_H$, $xx9A_H$

Function:

This function starts the programming process for one page as the normal “Write Page” does. But before programming it checks if the page is erased. If the page is not erased (allowing correctable errors) the command fails with PVER and EVER.

When the programming itself incurs an error the flag PVER is set.

On sectors with “write-once” protection only this write command can be applied.

This command is only supported for PFlash. When applied to DFlash the command fails with SQER.

Write Burst

Calling:

- ST $AA50_H$, PA
- ST $AA58_H$, $xx00_H$
- ST $AAA8_H$, $xxA0_H$
- ST $AAA8_H$, $xx7A_H$

Function:

This function starts the programming process for an aligned group of pages. The programming process is more efficient than for a single page, achieving a significantly higher throughput (see data sheet).

In the PFlash 8 pages (256 bytes) are programmed and in DFlash 4 (32 bytes). The pages are programmed one after the other with increasing addresses.

Please note that with disabled ECC generation with ECCW.PECENCDIS or DECENCDIS this command sequence must not be used. It causes unpredictable results.

For further details see “Write Page”.

Erase Logical Sector Range

Calling:

- ST $AA50_H$, SA
- ST $AA58_H$, $xxnn_H$
- ST $AAA8_H$, $xx80_H$
- ST $AAA8_H$, $xx50_H$

Function:

This command erases “nn” sectors starting at the sector addressed by “SA”. Upon entering command mode the BUSY flag of the corresponding bank is set. When erasing PFlash sectors additionally D0BUSY is set (because the erase counters are programmed in DF0).

SQER is returned when SA does not point to the base address of a correct sector (as specified at the beginning of this section) or to an unavailable sector.

When SA or any of the following nn-1 sectors has an active write protection or the Flash module has an active global read protection the execution fails and PROER is set.

The command fails with SQER if the range of logical sectors is not contained in one physical sector.

The error flag EVER is set to indicate an error during the erase process. The flags EVER and PVER are set when the erase counter programming incurs an error.

Note: The duration of an erase command applied to DFlash can be much shorter than documented as maximum duration (t_{ERD}, t_{MERD}).

Verify Erased Logical Sector Range

Calling:

- ST $AA50_H$, SA
- ST $AA58_H$, $xxnn_H$
- ST $AAA8_H$, $xx80_H$
- ST $AAA8_H$, $xx5F_H$

Function:

This command verifies if “nn” sectors starting at the sector addressed by “SA” are correctly erased, i.e. contain 0 data and ECC bits. Upon entering command mode the BUSY flag of the corresponding bank is set.

When the PFlash Safety ECC is enabled (see **Chapter 11.5.6.2**) the PFlash cannot be checked for “0” content by direct reads.

The error flag EVER is set to identify a found verification error.

SQER is returned when SA does not point to the base address of a correct sector (as specified at the beginning of this section) or to an unavailable sector.

The command fails with SQER if the range of logical sectors is not contained in one physical sector.

Especially security concerned customers can limit this function by configuring PROCONHSMCOTP.BLKFLAN to ‘1’:

- If PROCONHSMCOTP.BLKFLAN is 0 then “Verify Erase Logical Sector Range” is enabled for every address.
- If PROCONHSMCOTP.BLKFLAN is 1 then “Verify Erase Logical Sector Range” is blocked as follows:
  - If one of the PROCONCOTP.HSMsX = 1: blocked on the corresponding HSM code sector “s”.
  - If FPRO.PROINP = 1 and FPRO.PRODISP = 0: blocked on UCB_PFlash.
  - If FPRO.PROIND = 1 and FPRO.PRODISD = 0: blocked on UCB_DFlash.
  - If FPRO.PROINDBG = 1 and FPRO.PRODISDBG = 0: blocked on UCB_DBG.

If this function is blocked the command fails by setting PROER.

Resume Prog/Erase

Calling:

- ST $AA50_H$, PA/SA
- ST $AA58_H$, $xxnn_H$
- ST $AAA8_H$, $xx70_H$
- ST $AAA8_H$, $xxCC_H$

Function:

A suspended programming or erasing process is restarted as described in **Chapter 11.5.4.5**.

Disable Protection

Calling:

- ST $553C_H$, UC
- ST.W $553C_H$, PW0
- ST.W $553C_H$, PW1
- ST.W $553C_H$, PW2
- ST.W $553C_H$, PW3
- ST.W $553C_H$, PW4
- ST.W $553C_H$, PW5
- ST.W $553C_H$, PW6
- ST.W $553C_H$, PW7

Function:

The password protection of the selected UCB (if this UCB offers this feature) is temporarily disabled by setting FPRO.PRODISx (with “x” indicating the UCB) when all the passwords PW0–PW7 match their configured values in the corresponding UCB.

The command fails by setting PROER when any of the supplied PWs does not match. In this case until the next application reset all further calls of “Disable Protection” fail with PROER independent of the supplied password.

Resume Protection

Calling:

- ST 5554_H, $xxF5_H$

Function:

This command clears all FPRO.PRODISx effectively enabling again the Flash protection as it was configured.

Clear Status

Calling:

- ST 5554_H, xxFA_H

Function:

The flags FSR.PROG and FSR.ERASE and the error flags of **FSR** (OPER, SQER, PROER, PFSBER, PFDBER, PFMBER, DFSBER, DFDBER, DFTBER, DFMBER, ORIER, PVER, EVER) are cleared. These flags can be also cleared in the status registers without command sequence.

11.5.4.5 Operation Suspend and Resume

The operations “Erase Logical Sector Range”, “Verify Erased Logical Sector Range” and “Write Page”[1] and “Write Burst”[2] can be suspended.

An operation is suspended by writing ‘1’ to **MARD**.SPND. The Flash operation stops when it reaches an interruptible state. After that the flag **FSR**.SPND is set and busy is cleared.

There can be at most one operation in the suspended state. Erroneous suspend and resume requests are detected and prevented by the PMU.

The target range of a suspend program or erase operation is in an undefined state. Reading this range delivers unpredictable results.

The suspended operation can be resumed with the command sequence “Resume Prog/Erase”. The flag **FSR**.SPND is cleared and the busy flag is set again.

The arguments “PA” or “SA” and “nn” of the resume command must be identical to the original argument of the suspended command[3]. Using a different argument is detected by the PMU and lets the resume fail with FSR.SQER and FSR.SPND stays set.

When the operation had already finished when receiving the suspend request the request is ignored. The flag **FSR**.SPND is not set. Also **MARD**.SPND is immediately cleared again.

If a resume is requested without any suspended operation SQER is set.

Attention: Please ensure that between start or resume of an erase process and the suspend request normally at least ~1 ms erase time can pass. Shorter erase durations are allowed but the erase process will not advance.

In the suspended programming state:

1) The “Write Page” operation is suspended between end of the programming and start of the verification. Therefore the programming process itself is not suspended.

2) The “Write Burst” operation in PFlash is suspended during the programming process. In DFlash it is only suspended after end of the programming before starting the verification.

3) For resuming a “Write Burst” the argument “nn” of “Resume Prog/Erase” has to be “00”.

- Reading Flash is allowed.
- New programming and erase commands are rejected with SQER.

In the suspended erase state:

- Reading Flash is allowed.
- New erase commands are rejected with SQER.
- Programming commands can be performed on any bank.
- However programming in the target range of the suspended erase fails with SQER.
- Suspending a programming command is not possible and fails by setting **MARD**.SPNDERR.

In the suspended “Verify Erased Logical Sector Range” state:

- Reading Flash is allowed.
- New erase commands are rejected with SQER.
- Programming commands can be performed on any bank.
- Suspending a programming command is not possible and fails by setting **MARD**.SPNDERR.

11.5.4.6 Programming Voltage Selection

In devices which support 5 V supply via V_{EXT} this control bit allows to select the external supply as source of the programming voltage.

Attention: In this 3.3 V device this function is not supported and FCON.PR5V must be set to “P3V”.

11.5.5 Protection

Protection has several aspects: security, safety, preventing maloperation.

The safety protection ensures that reading from PFlash can’t be disturbed by software crashes or accesses by “unsafe” masters. Further on it ensures integrity of the read data. This is achieved by the following features:

- Safety ENDINIT protection of configuration registers.
- Master specific protection of PFlash relevant control and configuration registers.
- Protection of these registers against single-event effects “SEEs”.
- Master specific access control for command sequences.
- Integrity of read data is ensured by ECC (see **Chapter 11.5.6**).

The security protection prevents unauthorized Flash read and write from internal masters or from external:

- Flash read protection: protected Flash sections can only be read when booting from internal Flash.
- Flash write protection: protected Flash sections can only be changed after authentication.

- Flash OTP (One-Time Programmable) protection: these Flash sections can't be changed anymore.
- HSM specific protection: it ensures that the HSM can protect its private data from access by other software. Additionally selected data can only be read during HSM boot phase.
- HSM debug protection: when the HSM allows internal debugging, accesses by the debug master ("Cerberus") have the same privileges as the HSM itself.
- Device debug protection (see **Chapter 11.5.5.6**): depending on the boot mode and the configuration of debug protection and the Flash read protection and further controlled by HSM the debug access to the device is enabled.

The following features protect the Flash against maloperation including software crashes (even of "safe" masters):

- ENDINIT and SV mode protection of configuration registers.
- Command sequences that can change Flash content need more command cycles. SQER stops command interpretation.

Security features supplement safety because Flash data protected by the write protection is also safe against software crashes.

The following table gives an overview of all protection features and for which resources they are effective.

Table 11-7 Protection Layers

	PFlash	DFlash	Registers
Master Specific Access Control	–	–	Restricts Write Accesses
ENDINIT, SV, "Safety ENDINIT"	Restricts Commands (only SE)	–	Restrict Write Accesses
Flash Write Protection	Restricts Commands	Restricts Commands	–
Flash Read Protection	Restricts Read	Restricts Read	–

Any access to these resources has to pass all checks before the operation is performed.

Note: This means that an activation of a protection after start of the operation doesn't affect the operation anymore. Especially the "Safety ENDINIT" can become active as soon as a Flash command (e.g. program or erase) has started.

11.5.5.1 Master Specific Access Control

The master specific access control is configured and controlled by the registers ACCEN0 and ACCEN1 (see **Chapter 11.7.2.1**).

The registers ACCEN determine which master is allowed to perform writes to the protected resources. In the PMU the following resources are protected by this mechanism:

- Write access to all PMU and Flash registers with following exceptions (see also “Registers Overview” tables **Table 11-16** and **Table 11-18**):
  - Write access to ACCEN is controlled by the Safety ENDINIT.
  - Write accesses to the HSM private registers: HSMFSR, HSMMARD.

Write accesses that are blocked by this mechanism create a bus error as response.

11.5.5.2 Register Access Control

The master unspecific register access control features “Safety ENDINIT”, “ENDINIT”, “Supervisor Mode” are described in the SCU chapter.

In the PMU the following resources are protected by Safety ENDINIT:

- When the Safety ENDINIT is active program and erase commands on the PFlash are rejected with PROER.
- Write access to ACCEN.

11.5.5.3 Effective Flash Read Protection

The Flash read protection depends on the master and the Flash address range.

The following Flash ranges have separate read protections:

- PFlash (with specific handling of the HSM code sectors in PF0).
- And in the DFlash:
  - EEPROMs.
  - UCBs.

PFlash Read Protection

A read access to PFlash (with the exception of sectors S6, S16 and S17 in PF0 of PMU0) fails with bus error under the following conditions:

- Tricore CPU0 PMI (code fetch): FPRO.DCFP.
- Tricore CPU0 DMI (data read): FPRO.DDFP.
- All other masters: FPRO.DDFPX.

The read access to the HSM code sectors S6, S16 and S17 depends on the setting of PROCONHSMCOTP.HSMsX which defines the “HSM_exclusive” attribute of these sectors.

If PROCONHSMCOTP.HSMsX is set for such a sector:

- HSM: full read access.
- All masters except HSM: a read fails with a bus error.
- Cerberus: with enabled HSM debug same access rights as HSM itself, with disabled HSM debug access rights as all other masters.

If PROCONHSMCOTP.HSMsX is cleared for such a sector: the same rules as for the other PFlash0 sectors based on PROCONP0 as described above are valid.

DFlash Read Protection

A read access to the DFlash sectors EEPROMx fails with a bus error under the following conditions:

- All masters: FPRO.DDFD.

The read access to the UCBs is described in **Chapter 11.5.5.5**.

Activating Read Protection

The bits FPRO.DCFP, FPRO.DDFP, FPRO.DDFPX and FPRO.DDFD are initialized by the startup software depending on the configured protection and the startup mode. They can also be directly modified by the user software under conditions noted in the description of **FPRO**.

11.5.5.4 Effective Flash Write Protection

A range of Flash can be write protected by several means that are all configured in the UCBs.

PFlash Write Protection

Programming or erasing of a group of PFlash sectors (with the exception of sectors S6, S16 and S17 in PF0 of PMU0) fails with PROER if any of the following conditions is true:

- PROCONP0.RPRO and not FPRO.PRODISP (global write protection).
- PROCONPx.SxL and not FPRO.PRODISP (sector specific write protection).
- PROCONOTPx.SxROM (sector specific OTP protection).
- PROCONWOPx.SxWOP (sector specific write-once protection).
  - A set PROCONWOPx.SxWOP prohibits “Write Page”, “Write Burst” and erase commands. These fail with PROER.
  - Only “Write Page Once” is accepted by the PMU. The FSI checks if the addressed Flash range is erased. If this is not the case the FSI reports a PVER and doesn’t start programming.

Programming and erasing of the HSM reserved code sectors S6, S16 and S17 in PF0 of PMU0 depends on the setting of PROCONHSMCOTP.HSMsX which defines the “HSM_exclusive” attribute of these sectors and which master performs the access.

Programming or erase of such sectors fails if any of the following conditions is true:

- Not PROCONHSMCOTP.HSMsX and PROCONP0.RPRO and not FPRO.PRODISP (“global” write protection).
- Not PROCONHSMCOTP.HSMsX and PROCONP0.SxL and not FPRO.PRODISP (sector specific write protection).
- PROCONHSMCOTP.SxROM (HSM code sector specific OTP protection).
- For all masters except HSM: PROCONHSMCOTP.HSMsX.

For Cerberus the following rules apply: with enabled HSM debug same access rights as HSM itself, with disabled HSM debug access rights as all other masters.

DFlash Write Protection

Programming and erasing of the DFlash sectors EEPROMx fails with PROER when any of the following conditions is true:

- PROCOND.RPRO and not FPRO.PRODISD.
- PROCOND.L and not FPRO.PRODISD.

Program and erase conditions for the UCBs are described in **Chapter 11.5.5.5**.

11.5.5.5 Configuring Protection in the UCB

As indicated above the effective protection is determined by the content of the PROCONx registers. These are loaded during startup after each reset from the UCBs. Each UCB comprises 1 KByte of DFlash organized in 128 UC pages of 8 bytes.

An UCB is erased with the command “Erase Logical Sector Range”. It can be programmed with “Write Page” and “Write Burst”. Each UCB has its own access control.

When programming or erasing fail because of this access control PROER is set. When reading fails a bus error is returned.

The following UCBs are defined:

UCB_PFlash

Password protection of the PFlash.

Table 11-8 UCB_PFlash Content

Offset[1)]	Content	Description
00_H	PROCONP0	Read protection for complete PFlash. Write protection for logical sectors of PF0.
04_H		Reserved for PROCONP1.
08_H		Reserved for PROCONP2.

Table 11-8 UCB_PFlash Content (cont'd)

Offset[1)]	Content	Description
$0C_H$		Reserved for PROCONP3.
10_H	PROCONP0	Copy.
14_H		Reserved for copy of PROCONP1.
18_H		Reserved for copy of PROCONP2.
$1C_H$		Reserved for copy of PROCONP3.
20_H	PW0 – PW7	256-bit password, from least significant word to most significant word (8 words).
40_H	PW0 – PW7	Copy of 256-bit password.
70_H	Confirmation	4 bytes.
78_H	Confirmation	Copy.

1) The offsets are given in number of bytes.

See **PROCONPp (p=0-0)** on **Page 11-68**.

The confirmed state of this UCB is indicated by FPRO.PROINP.

In the errored state of this UCB also FPRO.PROINP is set. Disabling the protection is not possible in this state.

This UCB is write and read protected if:

- FPRO.PROINP and not FPRO.PRODISP.

UCB_DFlash

Password protection of the DFlash.

This UCB is also used for configuring the PFlash ECC algorithm between the "Safety" and the "Legacy" version, configuring the memory initialization and contains an oscillator configuration.

Table 11-9 UCB_DFlash Content

Offset[1)]	Content	Description
00_H	PROCOND	Read and write protection for the DFlash sectors EEPROMs, oscillator configuration, and enable of RAM initialization.
10_H	PROCOND	Copy.
20_H	PW0 – PW7	256-bit password, from least significant word to most significant word (8 words).

Table 11-9 UCB_DFlash Content (cont'd)

Offset[1)]	Content	Description
40_H	PW0 – PW7	Copy of 256-bit password.
70_H	Confirmation	4 bytes.
78_H	Confirmation	Copy.

1) The offsets are given in number of bytes.

See **PROCOND** on **Page 11-69**.

The confirmed state of this UCB is indicated by FPRO.PROIND.

In the errored state of this UCB also FPRO.PROIND is set. Disabling the protection is not possible in this state.

This UCB is write and read protected if:

- FPRO.PROIND and not FPRO.PRODISD.

UCB_OTP

This UCB configures the one-time programmable "OTP" and the write-page once "WOP" protection. It offers the possibility to add this type of protection to Flash sections incrementally.

Table 11-10 UCB_OTP Content

Offset[1)]	Content	Description
00_H	PROCONOTP0	OTP protection for the logical sectors of PF0, and enable of TP.
04_H		Reserved for PROCONOTP1.
08_H		Reserved for PROCONOTP2.
$0C_H$		Reserved for PROCONOTP3.
10_H	PROCONOTP0	Copy.
14_H		Reserved for copy of PROCONOTP1.
18_H		Reserved for copy of PROCONOTP2.
$1C_H$		Reserved for copy of PROCONOTP3.
20_H	PROCONWOP 0	WOP protection for the logical sectors of PF0.
24_H		Reserved for PROCONWOP1.
28_H		Reserved for PROCONWOP2.
$2C_H$		Reserved for PROCONWOP3.

Table 11-10 UCB_OTP Content (cont'd)

Offset[1)]	Content	Description
30_H	PROCONWOP 0	Copy.
34_H		Reserved for copy of PROCONWOP1.
38_H		Reserved for copy of PROCONWOP2.
$3C_H$		Reserved for copy of PROCONWOP3.
70_H	Confirmation	4 bytes.
78_H	Confirmation	Copy.
80_H		Start of next set.
…		

1) The offsets are given in number of bytes.

See **PROCONOTPp (p=0-0)** on **Page 11-71** and **PROCONWOPp (p=0-0)** on **Page 11-73**.

This configuration set can be stored 8 times. Each set has its own confirmation code. A protection set can be programmed when it is in the "unlocked" state, else it is write-protected.

The content of the PROCONOTP and PROCONWOP registers is initialized by the first configuration set at offset 0 (if this is "unlocked" or "confirmed"). All following confirmed sets are or-ed to this initial value.

An errored set activates all protection bits and activates write protection for this complete UCB.

Attention: This means that always each of the 8 confirmation codes and their copies shall be either in the "confirmed" or "unlocked" state (see UCB Confirmation on Page 11-36).

The confirmed state of the complete UCB is indicated by FPRO.PROINOTP. It is set when there is at least one protection set in the confirmed state.

If any set is in the errored state FPRO.PROINOTP is also set.

This UCB is protected from erasing if:

- FPRO.PROINOTP.

This UCB is readable.

UCB_HSMCFG

This UCB contains data supplied by Infineon during device production for the exclusive use of the HSM module.

UCB_IFX

This UCB contains data supplied by Infineon during device production.

Table 11-11 UCB_IFX Content

Offset[1]	Content	Description
00_H	UID	128-bit Unique ID.
10_H	Reserved	Reserved.
80_H		Up to 128 bytes used for DSADC trimming (see DSADC chapter in devices with DSADC).
100_H		Reserved for calibration data.
200_H		Reserved for test information data.
380_H		4 bytes “Test Pass Marker”.

1) The offsets are given in number of bytes.

This UCB is read-only. It is readable by every master.

The unique ID is a device unique 128-bit number. Its least significant byte has the value 64_D which identifies an Infineon device.

Note: In order to ensure proper traceability in case of FARs, the unique ID shall be stored by the customer in an adequate database.

The unique ID is supplemented with identification data in the SCU registers CHIPID (device identification and revision, speed grade, Flash size and µCode revision) and MANID (manufacturer identification). See SCU chapter.

These SCU registers are also sufficient to identify the needed Flash driver (replacement for the QRY record of the CFI standard).

The 4 bytes “Test Pass Marker” contains during test the value $FFFFFFFF_H$. When all tests have passed it signals this with the value 80658383_H. If customers receive a productive device with a different value in this address they can discard it.

UCB_DBG

This UCB configures the password protection for the debug interface.

Table 11-12 UCB_DBG Content

Offset[1]	Content	Description
00_H	PROCONDBG	Protection of the debug interface.
10_H	PROCONDBG	Copy.

Table 11-12 UCB_DBG Content (cont'd)

Offset[1)]	Content	Description
20_H	PW0 – PW7	256-bit password, from least significant word to most significant word (8 words).
40_H	PW0 – PW7	Copy of 256-bit password.
70_H	Confirmation	4 bytes.
78_H	Confirmation	Copy.

1) The offsets are given in number of bytes.

See **PROCONDBG** on **Page 11-74**.

The confirmed state of this UCB is indicated by FPRO.PROINDBG.

In the errored state of this UCB also FPRO.PROINDBG is set. Disabling the protection is not possible in this state.

This UCB is write and read protected if:

- (FPRO.PROINDBG and not FPRO.PRODISDBG) or (PROCONHSMCOTP.DESTDBG = "destructive" and (FDEST = 0 or "not executing Startup Software")) or PROCONDBG.EDM = "debug entered".

UCB_HSMCOTP

This UCB configures the OTP and HSM_exclusive protection for the dedicated HSM code sectors S6, S16 and S17 of PF0 of PMU0. It offers the possibility to add this type of protection to Flash sections incrementally.

Table 11-13 UCB_HSMCOTP Content

Offset[1)]	Content	Description
00_H	PROCONHSMCOTP	OTP and HSM_exclusive protection for HSM code sectors and configuration of the HSM boot.
10_H	PROCONHSMCOTP	Copy.
70_H	Confirmation	4 bytes.
78_H	Confirmation	Copy.
80_H		Start of next set.
…		

1) The offsets are given in number of bytes.

See **PROCONHSMCOTP** on **Page 11-76**.

This configuration set can be stored 2 times. Each set has its own confirmation code.

The content of the PROCONHSMCOTP register is initialized by the first configuration set at offset 0 (if this is “unlocked” or “confirmed”). All following confirmed sets are or-ed to this initial value.

The confirmed state of the complete UCB is indicated by FPRO.PROINHSMCOTP. It is set when there is at least one protection set in the confirmed state.

If any set is in the errored state FPRO.PROINHSMCOTP is also set. An errored set activates all protection bits.

Attention: This means that always each of the 2 confirmation codes and their copies shall be either in the “confirmed” or “unlocked” state (see UCB Confirmation on Page 11-36).

This UCB is protected from erasing when:

- FPRO.PROINHSMCOTP is set.

The set at offset 0 is protected from programming for all masters when it is in confirmed or errored state.

The set at offset 80_H is protected from programming for all masters except HSM when:

- FPRO.PROINHSMCOTP is set.

It is protected from programming by HSM when it is in the confirmed or errored state[1)].

This UCB is readable.

With enabled HSM debug Cerberus has the same access rights as HSM itself. With disabled HSM debug Cerberus is treated as any other master.

UCB_HSM

This UCB can be programmed and erased by all CPUs. However when its content is confirmed or errored only the HSM can modify this UCB.

Its content is only used in devices with activated HSM.

With enabled HSM debug Cerberus has the same access rights as HSM itself. With disabled HSM debug Cerberus is treated as any other master.

Table 11-14 UCB_HSM Content

Offset[1)]	Content	Description
00_H	PROCONHSM	HSM Configuration.
10_H	PROCONHSM	Copy.

1) Rationale: all masters can configure the protection in delivery state (both sets unlocked). When the first set is confirmed only HSM is allowed to program the second set. When this is also confirmed the complete UCB is OTP protected.

Table 11-14 UCB_HSM Content (cont'd)

Offset[1)]	Content	Description
70_H	Confirmation	4 bytes.
78_H	Confirmation	Copy.

1) The offsets are given in number of bytes.

See **PROCONHSM** on **Page 11-79**.

The confirmed state of this UCB is indicated by FPRO.PROINHSM.

In the errored state of this UCB also FPRO.PROINHSM is set.

This UCB is protected from programming and erasing by non-HSM masters if:

- FPRO.PROINHSM.

It can be read by every master.

Erase Counter UCB12 – UCB15

These UCBs are used to implement erase counters. From user point of view their content is read-only. They are readable by every master.

For every call of an "Erase Logical Sector Range" the FSI increases automatically the corresponding erase counters until the maximum counting range is reached.

Each UCB12 – UCB15 contains the erase counters for one bank of PFlash PF0 – PF3.

Each logical sector Sx of one PFlash bank has an assigned UCB range of 32 bytes starting at offset address x * 32_D.

The counting is performed by programming the least significant unprogrammed nibble with 1111_B. For reliability reasons a nibble is considered as "unprogrammed" when it contains at most one 1-bit.

Each counter saturates when the last nibble is programmed. Therefore each counter saturates after 64_D erases.

For evaluating the counters the DFlash ECC correction must be disabled with ECCRD.ECCORDIS because this "thermometer code" is not ECC correct.

Remaining UCB8 – UCB11

Currently not used.

These UCBs are read-only.

UCB Confirmation

As described above the UCBs (or sets of a UCB in case of UCB_OTP and UCB_HSMCOTP) can be in the states “confirmed”, “unlocked” and “errored”. This state is determined by the content of the confirmation code:

- 57B5 $327F_H$: the state is “confirmed”.
- 4321 1234_H: the state is “unlocked”. For over-programming this state (see below) the 4 bytes following the confirmation code need to be kept as 0.
- all other content including uncorrectable ECC errors: the state is “errored”.

An UCB is also in its errored state when at least one of its entries has an uncorrectable error in the original and its copy.

When during evaluation of the UCBs an errored state is detected the flag FSR.PROER is set. The derived values in the PROCONx registers are not set to the Flash content but as indicated in the register description the protection is fully activated. When during startup an errored UCB is detected by the SSW it recognizes this as failed Flash startup and consequently doesn’t boot the device.

As also the erased state is considered as “errored” the transition from “unlocked” to “confirmed” state can be done without erasing the UCB. For this the “unlocked” code in the pages with the confirmation code and the copied confirmation code can be over-programmed with 57B5 $327F_H$. It has to be ensured that the 4 bytes following the confirmation code (e.g. at offset 74_H, $7C_H$) are kept 0000 0000_H in the unlocked state and in the over-programmed data. Only then the result after over-programming is ECC clean.

Attention: In case of the multi-set UCBs “UCB_OTP” and “UCB_HSMCOTP” each of the n (8 or 2) confirmation codes and their copies have to be in the state “unlocked” or “confirmed” to prevent an errored state.

11.5.5.6 System Wide Effects of Flash Protection

An active Flash read protection needs to be respected in the complete system.

The startup software “SSW” checks if the HSM is available.

If yes the HSM module is booted. During its boot process it can lock the device debug interface.This interface can be locked either by HSM (see below **HSM Booting**) or by setting OSTATE.IF_LCK[1)].

Additionally the customer can configure in UCB_DBG that OCDS or the debug interface are disabled.

This results in the following lock conditions:

- OCDS disabled: PROCONDBG.OCDSDIS and not FPRO.PRODISDBG.
- Debug interface locked: OSTATE.IF_LCK[2)] or (HSM lock input) or (PROCONDBG.DBGIFLCK and not FPRO.PRODISDBG.

1) The register OSTATE is contained in the OCDS module. The distribution of OCDS documentation is restricted.

The SSW performs the following operations:

- The SSW leaves the debug interface locked (OSTATE.IF_LCK stays 1) if any Flash read protection is configured (PROCONP0.RPRO or PROCOND.RPRO are 1).
- If the selected boot mode executes from internal PFlash:
  – The SSW clears the FPRO bits DCFP, DDFP, DDFPX, DDFD.
- If the selected boot mode does not execute from internal PFlash:
  – The SSW sets the FPRO bits DCFP, DDFP, DDFPX, DDFD if their corresponding read protection is configured in PROCONP0.RPRO or PROCOND.RPRO.

HSM Booting

The SSW boots the HSM module (i.e. HSM executes its firmware) when the field PROCONHSMCOTP.HSMBOOTEN is set (i.e. the customer has installed HSM boot code in the HSM code sectors).

First the HSM firmware checks PROCONHSMCOTP.HSMDBGDIS. If this is cleared it enables HSM debug.

After that the HSM firmware checks PROCONHSM.DBGIFLCK. If this is cleared it releases its lock of the system debug interface.

After execution of the HSM firmware the HSM executes from the HSM code sectors (see HSM chapter). Depending on PROCONHSMCOTP.SSWWAIT the SSW waits for an acknowledge from the HSM before leaving the Firmware.

When the execution of the HSM firmware fails (e.g. due to missing HSM code) this is signaled to the SSW which stops executing. Consequently the device doesn't boot into user mode.

Destructive Debug Entry

As described before the debug interface can be opened by supplying the correct password of UCB_DBG to the PMU under the condition that HSM doesn't lock this interface.

A special destructive debug entry can be configured with **PROCONHSMCOTP**.DESTDBG. When this is configured as "destructive" only the SSW can accept the UCB_DBG password via debug interface (see BootROM chapter). Additionally the password is only accepted when the device pin "FDEST" is logic '1' and the field **PROCONDBG**.EDM is "debug not entered". When all these conditions match the field **PROCONDBG**.EDM is programmed by the SSW to "debug entered" before opening the debug interface. When EDM is "debug entered" automatically the clock system is manipulated to disallow communication via CAN or Flexray.

2) Only for illustration, this bit and its or-combination with the following signals is part of the OCDS not the PMU. The distribution of OCDS documentation is restricted.

11.5.6 Data Integrity and Safety

The data in Flash is stored with error correcting codes “ECC” in order to protect against data corruption. Also data transfer over the SRI bus is protected with ECC. The healthiness of Flash data can be checked with margin checks.

The following measures ensure for reading from PFlash a single point failure metric of >99% and a latent fault metric of >90%.

11.5.6.1 SRI ECC (Safe Fetch Path)

The data transfer over the SRI bus is protected by ECC. This is described in the Safety Concept. The address phase signals are accompanied with an ECC code with Hamming distance 4. The write and read data phase signals are accompanied with an ECC code (also calculated over the address) with Hamming distance 4.

11.5.6.2 Flash ECC

The data in Flash is stored with ECC codes. These are automatically generated when the data is programmed. When data is read these codes are evaluated. Depending on the Flash bank different algorithms with different error correction capabilities are used:

- Data in PFlash uses an ECC code with DEC-TED (Double Error Correction, Triple Error Detection) capabilities. Each block of 256 data bits is accompanied with a set of ECC bits.
  - The ECC algorithm in the PFlash can be switched between a “Safety ECC” which is default and a “Legacy ECC”.
  - The selection of this algorithm is done by the register field **FCON**.NSAFECC. This is set by the startup software depending on **PROCOND**.NSAFECC but can also be modified by software.
- Data in DFlash uses an ECC code with TEC-QED (Triple Error Correction, Quad Error Detection) capabilities. Each block of 64 data bits is accompanied with a set of ECC bits.

PFlash Safety ECC Details

The PFlash Safety ECC value is calculated over 256 data bits and the address bits 22:5 and the configuration area selection signal. This ECC has the following features:

- Correction of 1-bit and 2-bit errors.
- Detection of 3-bit errors.
- Detection of >99% of all error vectors in the white noise error model.
- Detection of >99% of all-0 and all-1 cases.
- Detection of addressing errors.

As side effect of the all-0 error detection an erased Flash range can't be read without ECC errors. Also over-programming of Flash ranges with all-1 would create entries with ECC errors.

The ECC is automatically generated when programming the PFlash when this is not disabled with ECCW.PECENCDIS.

The ECC is automatically evaluated when reading data. Errors are only reported for 256-bit data blocks for which at least one byte is read by a bus master. Errors found in internally pre-fetched data is stored and only reported when this data block is requested by a bus master.

The PMU does not know if the bus master uses the requested data or discards it (e.g. due to speculatively fetching code). Therefore this speculatively fetched data gets the same error response.

Error reporting and ECC disabling:

- Single-bit error:
  - Is noted in FSR.PFSBER.
  - Is reported to the SMU (which can trigger an interrupt).
  - Data and ECC value are corrected if this is not disabled with ECCRPp.ECCORDIS[1].
  - Depending on CBABCFGp.DIS and CBABCFGp.SEL the affected address is stored uniquely in the CBAB.
- Double-bit error:
  - Is noted in FSR.PFDBER.
  - Is reported to the SMU (which can trigger an interrupt).
  - Data and ECC value are corrected if this is not disabled with ECCRPp.ECCORDIS[1].
  - Depending on CBABCFGp.DIS and CBABCFGp.SEL the affected address is stored uniquely in the CBAB.
- Triple-bit error (i.e. uncorrectable bit error):
  - Is noted in FSR.PFMBER.
  - Is reported to the SMU (which can trigger an interrupt).
  - Causes a bus error if not disabled by MARP.TRAPDIS.
  - Depending on UBABCFG.DIS and UBABCFG.SEL the affected address is stored in the UBAB.
  - Can additionally report an address error.
- Address error:
  - Is noted in FSR.PFMBER.
  - Is reported to the SMU (which can trigger an interrupt).
  - Causes a bus error if not disabled by MARP.TRAPDIS.

1) Disabling the correction of errors with ECCORDIS is a test feature. During application run-time the correction must be enabled.

  - Depending on UBABCFG.DIS and UBABCFG.SEL the affected address is stored in the UBAB.
- All-0 and all-1 errors:
  - Are noted in FSR.PFMBER[1)].
  - Is reported to the SMU (which can trigger an interrupt).
  - Causes a bus error if not disabled by MARP.TRAPDIS.
  - Depending on UBABCFG.DIS and UBABCFG.SEL the affected address is stored in the UBAB.

PFlash Legacy ECC Details

The “Legacy” PFlash ECC is calculated only over 256 data bits, not over address bits or the configuration selection. Therefore addressing faults (correct data is read from an incorrect address) can not be detected by this algorithm.

This algorithm has the advantage that an erased PFlash range delivers ECC correct 0 data.

Consequently all-0 data and ECC is not reported as an error.

Note: Please note that this algorithm (in contrast to the Audo generation) does not have all-1 ECC bits for all-1 data, i.e. over-programming a page with all-1 will not create an ECC clean data block.

The error detection, reporting and correction capabilities for single-bit, double-bit and triple-bit errors is identical to the Safety ECC.

DFlash ECC Details

The DFlash ECC value is calculated over 64 data bits. This ECC has the following features:

- Correction of 1-bit, 2-bit and 3-bit errors.
- Detection of 4-bit errors.
- All-0 (data and ECC) is a correct code vector.
- All-1 (data and ECC) is a valid code vector.
- Address errors are not detected.

An erased data block has an ECC value 0000_H. Therefore an erased sector is free of ECC errors. A data block with all bits ‘1’ has an ECC value of $FFFF_H$. This is commonly used in EEPROM emulations to over-program data to mark it as “invalid”.

Error reporting and ECC disabling:

- Single-bit, double-bit, triple-bit errors:
  - Are noted in FSR.DFSBER, FSR.DFDBER, FSR.DFTBER respectively.

1) On some addresses additionally FSR.PFDBER is set.

  - Data and ECC value are corrected if this is not disabled with ECCRD.ECCORDIS[1)].
- Quad-bit error:
  - Is noted in FSR.DFMBER.
  - Causes a bus error if not disabled by MARD.TRAPDIS.

11.5.6.3 Margin Checks

The Flash memory offers a “margin check feature”: the limit which defines if a Flash cell is read as logic ‘0’ or logic ‘1’ can be shifted. This is controlled by the registers **MARP** and **MARD**.

After changing the read margin at least $t_{FL_MarginDel}$ have to be waited before reading the affected Flash module.

11.5.6.4 PMU and Flash Register Supervision

Safety relevant registers of PMU and Flash are automatically protected against SEUs by the extensive ECC coverage of the complete PFlash read path.

Only the following configuration registers should be supervised by the safety software against accidental modification:

- FCON.STALL: However modification of this affects safety only in case of errors in the application software, when this is reading busy Flash banks.
- MARP.TRAPDIS: Not really safety relevant because all data read errors are anyhow notified to the SMU.

11.5.7 Interrupts and Traps

Generally the PMU reports fatal errors by issuing a bus error which is translated by the CPU into a trap (PMI sets PSE trap and DMI the DSE trap).

This is a list of conditions for reporting a bus error:

- Uncorrectable ECC error (if not disabled by MARP or MARD).
- Write access to read-only register.
- Write access to BootROM.
- Write access to an access controlled register or PFlash address range by a master without allowance by the register access protection (**Chapter 11.5.5.1**).
- Not allowed write access to protected register (e.g. SV or ENDINIT).
- Not allowed Flash read access with active read protection.
- Read and write access to all Flash memories when the Flash is in sleep mode.
- Read access to not available Flash memory.
- Read-modify-write access to the Flash memory.
- Read accesses to a busy Flash bank (if not disabled with **FCON**.STALL).

- Write access to DF0 address range when normal command interpreter is busy with a command.
- Read or write access to unoccupied register address.

Furthermore selected events can trigger interrupts. The service request nodes SRC_PMUp0/1 are documented in the IR[1].

The following events can trigger the SRC_PMUp0 service request node when enabled by **FCON**:

- End of busy: any transition of any of the FSR flags P0BUSY, P1BUSY, D0BUSY or D1BUSY from ‘1’ to ‘0’ triggers the interrupt (program and erase sequences, wake-up).
- Operational error: see OPER flag.
- Verify error: see PVER, EVER flags.
- Protection error: see PROER flag.
- Sequence error: see SQER flag.

The event that triggered the interrupt can be determined from the **FSR** register.

An interrupt event it also triggered when the event appears again and the corresponding status flag is still set.

The requested read feature (**Chapter 11.5.3.4**) signals availability of data with a separate read interrupt via SRC_PMUp1. This can be used to trigger a DMA controller.

The HSM command interface can trigger an application interrupt at the HSM module. The following events are notified when enabled by **HSMFCON**:

- End of busy: any transition from ‘1’ to ‘0’ of any of the **HSMFSR** busy flags.
- Operational error and verify error when any of the **HSMFSR** flags OPER, PVER, EVER gets set.
- Protection error when **HSMFSR**.PROER is set.
- Sequence error when **HSMFSR**.SQER is set.

11.5.8 Reset and Startup

The PMU is reset with the application reset with the exception of the following resources:

- Register bits: FSR.PROG, FSR.ERASE, FSR.OPER. These bits are reset with the power-on reset.
- The complete CBAB and UBAB (see **Chapter 11.7.2.13** and **Chapter 11.7.2.14**) including their status registers. These are also reset with the power-on reset.

The Flash module and the FSI are only reset with the system reset.

When during a Flash operation (program, erase) an application reset occurs the FSI is not reset but the operation is aborted to ensure that the Flash is readable during startup (see **Chapter 11.8.5**).

1) The parameter “p” in SRC_PMUp0/1 is “0” for PMU0 and “1” for PMU1.

During startup after the application reset the protection setting is installed from the UCBs. If a protection field has an uncorrectable error its copy is used. This is indicated by setting FSR.ORIER. If a copy has also an uncorrectable error the startup software does not enter the user code.

11.5.9 Power Reduction

The Flash module can enter sleep mode to reduce its power consumption. The sleep request can have two sources:

- Global sleep mode requested by the SCU (see “Power Management”). Only executed by the Flash when FCON.ESLDIS = 0.
- Writing ‘1’ to the bit FCON.SLEEP.

After receiving a sleep request the Flash starts the ramp down when the Flash becomes idle, i.e. none of the banks is in command mode and no reads are executed anymore. An ongoing read burst is finished completely. During ramp down to sleep mode all FSR.BUSYx are set.

The sleep mode is indicated in FSR.SLM. The FSR.BUSYx stay set.

The Flash module can be woken up by releasing the sleep request. It enters read mode again after t_{WU}. During the wake-up phase the FSR.BUSYx are set.

Note: It is not recommended to use the SCU controlled sleep mode with FCON.ESLDIS = 0. Because software had to ensure that upon wake-up the Flash is only read after it has returned to read mode. Earlier reads cause bus errors! Therefore the reset value of FCON.ESLDIS is 1.

Note: Requesting sleep mode does not disable automatically an enabled end-of-busy interrupt. When requesting sleep mode during an ongoing Flash operation with enabled end-of-busy interrupt, the operation finishes, the interrupt is issued and then the Flash enters sleep mode. However this end-of-busy interrupt will wake-up the CPU again.

An additional power reduction feature is enabled by setting FCON.IDLE. In this case the PFlash read path is switched off when no read access is pending and the read buffers are filled. System performance is slightly reduced because a flash line hit can not be exploited.

11.6 Signaling to the Safety Management Unit (SMU)

The Safety Management Unit “SMU” is described in its own chapter. This section describes the signaling from the PMU to the SMU.

The following error conditions are internally detected and reported to the SMU:

- PMU registers are automatically supervised by the extensive ECC coverage of the PFlash read path (see **Chapter 11.5.6.4**). No dedicate alarm is reported to the SMU.

- PFlash detected ECC errors. There are separate signals for corrected single-bit errors, corrected double-bit errors, address errors and uncorrected bit errors.
- CBAB became full.
- SRI ECC errors from address phase, data write phase, data read phase.

The ECC correction logic of each PFlash read path is continuously supervised by the “ECC Monitor”. A detected error is notified to the SMU.

The detection of this error can be checked by disabling the error correction with ECCRPp.ECCORDIS and reading ECC errored data from PFlash. As the correction is disabled the checker reports an error in the correction logic.

The ECC error detection logic of each PFlash read path is also independently continuously supervised by a redundant error detection unit and a comparator “EDC Comparator”. A detected error is notified to the SMU.

The detection of this error can be checked separately for each PFlash “p” error detection logic by setting bit ECCERRPp.EDCERRINJ. An error is injected in the checked detection path.

11.7 Register Set

The register set consists of some general PMU registers which are partly only implemented for PMU0 because they control its specific functionality (**Chapter 11.7.1**). The other registers control Flash functionality (**Chapter 11.7.2**). Register fields related to DFlash and its sub-sectors are implemented also when these Flash banks or their functionality is not available but they are without function.

The register access conditions use the following abbreviations:

- “U”: Access permitted in User Mode 0 or 1 (applicable to write and read).
- “SV”: Access permitted in Supervisor Mode (applicable to write and read).
- “E”: ENDINIT protected write. “E” means a write access is only allowed before ENDINIT or after disabling this protection with a password as described in the SCU chapter.
- “SE”: Safety ENDINIT protected write.
- “P”: The access is controlled by the master specific register access protection (**Chapter 11.5.5.1**).
- “H”: The access is only permitted for the HSM master or for Cerberus when HSM debug is enabled.
- “–” or “BE”: Access not permitted.

The PMU and Flash use the following combinations (see **Table 11-16** and **Table 11-18**):

- U, SV: access always allowed (i.e. in user mode or supervisor mode).
- P, U, SV: access always to masters allowed that are enabled by the register access protection (**Chapter 11.5.5.1**).
- P, SV: access only in supervisor mode for masters enabled by the register access protection.
- BE: access never allowed, causing a bus error.
- SV, E: access only in supervisor mode with disabled ENDINIT protection.
- P, SV, E: access only in supervisor mode with disabled ENDINIT protection for masters enabled by the register access protection.
- SV, SE: access only in supervisor mode with disabled Safety ENDINIT protection.

All accesses prevented due to these restrictions fail with a bus error.

Also accesses to unoccupied register addresses fail with a bus error.

Note: It is convention to use short register names (e.g. “FCON”) in the chapter that defines these registers. In all other chapters and in the development tools long register names are used that are a concatenation of the module instance (e.g. “PMU0” or “FLASH0”), an underscore and the short register name, i.e. “FLASH0_FCON”. This document uses for clarification also mostly the long names.

11.7.1 PMU Registers

Non-Flash related registers for the PMU. They have the prefix “PMUx_”.

Table 11-15 Registers Address Space

Module	Base Address	End Address	Note
PMU0	$F800\ 0500_H$	$F800\ 052B_H$	

Table 11-16 Registers Overview

Short Name	Description	Offset Addr.[1]	Access Mode		Reset	Page
			Read	Write		
ID	Module Identification	08_H	U, SV	BE	Application Reset	11-46

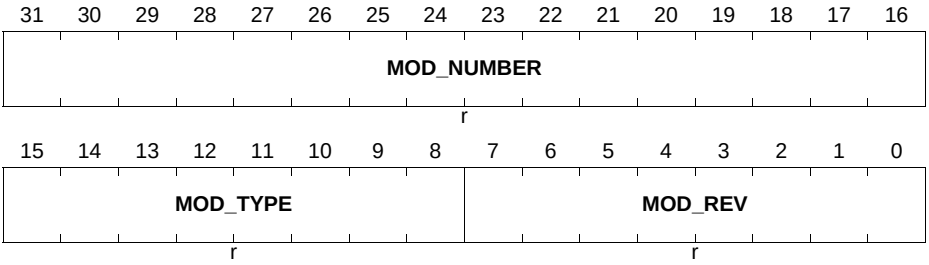
1) The absolute register address is calculated as follows:
Module Base Address (**Table 11-15**) + Offset Address (shown in this column)

11.7.1.1 PMU Identification

The PMU_ID register identifies the PMU and its version.

PMU0_ID

PMU0 Identification Register (F800 0508$_H$) Reset Value: 00D4 C0XX$_H$



Field	Bits	Type	Description
MOD_REV	[7:0]	r	**Module Revision Number** MOD_REV defines the module revision number. The value of a module revision starts with 01_H (first rev.).

Field	Bits	Type	Description
MOD_TYPE	[15:8]	r	**Module Type** This bit field is $C0_H$. It defines the module as a 32-bit module.
MOD_NUMBER	[31:16]	r	**Module Number Value** This bit field defines the module identification number for PMU0.

11.7.2 Flash Registers

The absolute address of a Flash register is calculated by the base address from **Table 11-17** plus the offset address of this register from **Table 11-18**.

Table 11-17 Registers Address Space

Module	Base Address	End Address	Note
FLASH0	F800 1000_H	F800 $23FF_H$	Flash registers of PMU0

Table 11-18 Registers Overview

Short Name	Description	Offset Addr.[1]	Access Mode		Reset	Page
			Read	Write		
COMM0	FSI Communication 0	0000_H	U, SV	P, SV	Application Reset	11-105
COMM1	FSI Communication 1	0004_H	U, SV	P, SV	Application Reset	11-105
COMM2	FSI Communication 2	0008_H	U, SV	P, SV	Application Reset	11-105
ID	Flash Module Identification Register	1008_H	U, SV	BE	Application Reset	11-52
FSR	Flash Status Register	1010_H	U, SV	P, U, SV	Application Reset [2]	11-53
FCON	Flash Configuration Register	1014_H	U, SV	P, SV, E	Application Reset	11-60
FPRO	Flash Protection Control and Status	$101C_H$	U, SV	P, SV, E	Application Reset	11-64
PROCONP0	PFlash Protection Config. PF0	1020_H	U, SV	BE	Application Reset	11-68
Res. PROCONP1	Reserved for PFlash Protection Config. PF1	1024_H	U, SV	BE	Application Reset	11-68
PROCOND	DFlash Protection Config.	1030_H	U, SV	BE	Application Reset	11-69
PROCONH SMCOTP	HSM Code Flash OTP Protection Config.	1034_H	U, SV	BE	Application Reset	11-76
PROCONO TP0	OTP Protection Config. PF0	1038_H	U, SV	BE	Application Reset	11-71

Table 11-18 Registers Overview (cont'd)

Short Name	Description	Offset Addr.[1)]	Access Mode		Reset	Page
			Read	Write		
PROCONWOP0	Write-Once Protection Config. PF0	1048_H	U, SV	BE	Application Reset	11-73
PROCONDBG	Debug Interface Protection Config.	1058_H	U, SV	BE	Application Reset	11-74
PROCONHSM	HSM Exclusive Config.	$105C_H$	U, SV	BE	Application Reset	11-79
RDBCFG00	Read Buffer Port 0 Configuration 0	1060_H	U, SV	P, SV, E	Application Reset	11-80
RDBCFG01	Read Buffer Port 0 Configuration 1	1064_H	U, SV	P, SV, E	Application Reset	11-80
RDBCFG02	Read Buffer Port 0 Configuration 2	1068_H	U, SV	P, SV, E	Application Reset	11-80
ECCW	ECC Write Register	1090_H	U, SV	P, SV, E	Application Reset	11-84
ECCRP0	ECC Read Register PF0	1094_H	U, SV	P, SV, E	Application Reset	11-85
ECCRD	ECC Read Register DF	$10A4_H$	U, SV	P, SV, E	Application Reset	11-85
MARP	Margin Control Register PFlash	$10A8_H$	U, SV	P, SV, E	Application Reset	11-95
MARD	Margin Control Register DFlash	$10AC_H$	U, SV	P, U, SV	Application Reset	11-96
CBABCFG0	Corrected Bits Address Buffer Configuration Port 0	$10B4_H$	U, SV	P, SV, E	Power-On Reset	11-98
CBABSTAT0	Corrected Bits Address Buffer Status Port 0	$10B8_H$	U, SV	BE	Power-On Reset	11-99
CBABTOP0	Corrected Bits Address Buffer Top Port 0	$10BC_H$	U, SV	P, SV	Power-On Reset	11-100
UBABCFG0	Uncorrectable Bits Address Buffer Configuration Port 0	$10E4_H$	U, SV	P, SV, E	Power-On Reset	11-101

Table 11-18 Registers Overview (cont'd)

Short Name	Description	Offset Addr.[1]	Access Mode		Reset	Page
			Read	Write		
UBABSTAT0	Uncorrectable Bits Address Buffer Status Port 0	$10E8_H$	U, SV	BE	Power-On Reset	11-102
UBABTOP0	Uncorrectable Bits Address Buffer Top Port 0	$10EC_H$	U, SV	P, SV	Power-On Reset	11-103
RRCT	Requested Read Control Register	1140_H	U, SV	U, SV	Application Reset	11-81
RRD0	Requested Read Data Register 0	1144_H	U, SV	BE	Application Reset	11-82
RRD1	Requested Read Data Register 1	1148_H	U, SV	BE	Application Reset	11-82
RRAD	Requested Read Address Register	$114C_H$	U, SV	U, SV	Application Reset	11-83
HSMFSR	HSM Flash Status Register	1200_H	H	H	Application Reset	11-86
HSMFCON	HSM Flash Configuration Register	1204_H	H	H	Application Reset	11-89
HSMMARD	HSM DFlash Margin Control Register	1208_H	H	H	Application Reset	11-90
HSMRRCT	HSM Requested Read Control Register	$120C_H$	H	H	Application Reset	11-92
HSMRRD0	HSM Requested Read Data Register 0	1210_H	H	BE	Application Reset	11-93
HSMRRD1	HSM Requested Read Data Register 1	1214_H	H	BE	Application Reset	11-93
HSMRRAD	HSM Requested Read Address Register	1218_H	H	H	Application Reset	11-94
ACCEN1	Access Enable Register 1	$13F8_H$	U, SV	SV, SE	Application Reset	11-51
ACCEN0	Access Enable Register 0	$13FC_H$	U, SV	SV, SE	Application Reset	11-51

1) The absolute register address is calculated as follows:
Module Base Address (**Table 11-17**) + Offset Address (shown in this column)

2) Some bits are not reset with the application reset but only with the power-on reset.

11.7.2.1 Master Specific Access Control

The master specific access control is described in **Chapter 11.5.5.1**.

Access Enable Register 0

The Access Enable Register 0 controls write access for transactions with the on-chip bus master TAG IDs 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID to master peripheral mapping). The PMU is prepared for a 6 bit TAG ID. The registers ACCEN0 and ACCEN1 provide one enable bit for each possible 6 bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000_B, EN1 -> TAG ID 000001_B, …, EN31 -> TAG ID 011111_B.

ACCEN0
Access Enable Register 0 ($13FC_H$) Reset Value: FFFF FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN 31	**EN 30**	**EN 29**	**EN 28**	**EN 27**	**EN 26**	**EN 25**	**EN 24**	**EN 23**	**EN 22**	**EN 21**	**EN 20**	**EN 19**	**EN 18**	**EN 17**	**EN 16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

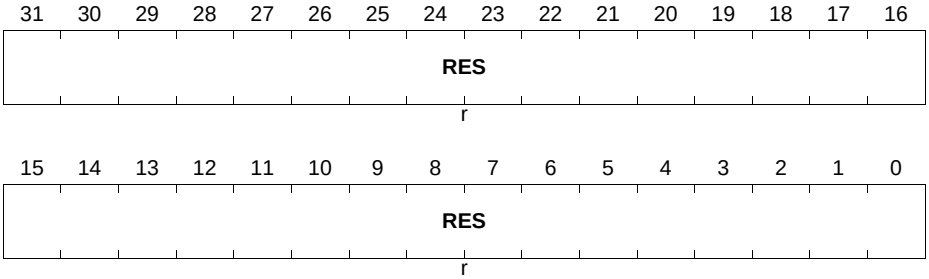
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN 15	**EN 14**	**EN 13**	**EN 12**	**EN 11**	**EN 10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENx (x = 0-31)	x	rw	**Access Enable for Master TAG ID x** This bit enables write access to the protected resources (see **Chapter 11.5.5.1**) for transactions with the corresponding Master TAG ID 0_B Write access will not be executed 1_B Write access will be executed

Access Enable Register 1

The Access Enable Register 1 controls write access for transactions with the on chip bus master TAG ID 100000_B to 111111_B (see On Chip Bus chapter for the products TAG ID to master peripheral mapping). The PMU is prepared for a 6 bit TAG ID. ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the AURIX devices.

ACCEN1
Access Enable Register 1 **(13F8$_H$)** **Reset Value: 0000 0000$_H$**



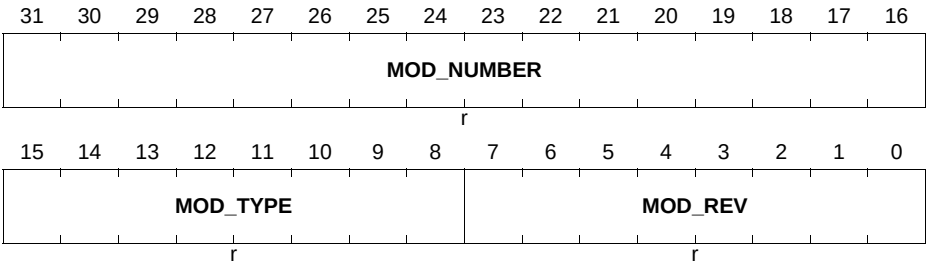
Field	Bits	Type	Description
RES	[31:0]	r	**Reserved** Read as 0; should be written with 0.

11.7.2.2 Flash Identification Register

The register identifies the Flash module which can have a different version from the PMU.

Flash Identification

ID
Flash Module Identification Register (1008$_H$) **Reset Value: 00DF (TC23x) / 00DE (TC22x) C0XX$_H$**



Field	Bits	Type	Description
MOD_REV	[7:0]	r	**Module Revision Number** MOD_REV defines the module revision number. The value of a module revision starts with 01_H (first revision).
MOD_TYPE	[15:8]	r	**Module Type** This bit field is $C0_H$. It defines the module as a 32-bit module.
MOD_NUMBER	[31:16]	r	**Module Number Value** This bit field defines a module identification number. For the TC21x/TC22x/TC23x Flash0 this number is 00DF (TC23x) / 00DE (TC22x)$_H$.

11.7.2.3 Flash Status

The Flash Status Register FSR reflects the overall status of the Flash module after Reset and after reception of the different commands.

The error flags and the two status flags (PROG, ERASE) are affected by the “Clear Status” command. The error flags are also cleared with the “Reset to Read” command.

Some error flags are marked as writable. These flags can be cleared by writing a ‘1’ to this bit.

Flash Status Register

FSR
Flash Status Register **(1010_H)** **Reset Value: $0100\ 0000_H$[1)]**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**ORIER**	**0**	**SLM**	**SPND**	**EVER**	**PVER**	**RES**	**0**	**SRIADDERR**	**DFMBER**	**DFTBER**	**DFDBER**	**DFSBER**	**RES17**	**PFMBER**
r	rh	r	rh	rwh	rwh	rwh	rwh	r	rwh	rwh	rwh	rwh	rwh	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PFDBER	**PFSBER**	**PROER**	**SQER**	**OPER**	**DFPAGE**	**PFPAGE**	**ERASE**	**PROG**	**RES6**	**RES5**	**RES4**	**P0BUSY**	**RES1**	**D0BUSY**	**FABUSY**
rwh	rwh	rwh	rwh	rwh	rh	rh	rwh	rwh	rh	rh	rh	rh	rh	rh	rh

1) The startup process might report errors by setting some of the error flags.

Field	Bits	Type	Description
FABUSY	0	rh	**Flash Array Busy** [1)] Internal busy flag for testing purposes. Must be ignored by application software. This may only use PxBUSY and DxBUSY.
D0BUSY	1	rh	**Data Flash Bank 0 Busy**[1)] HW-controlled status flag. 0_B DFlash0 ready, not busy; DFlash0 in read mode. 1_B DFlash0 busy; DFlash0 not in read mode. Indication of busy state of DFlash bank 0 because of active execution of an operation; DFlash0 busy state is also indicated during Flash startup after reset or in sleep mode; while in busy state the DFlash0 is not in read mode.
RES1	2	rh	**Reserved for Data Flash Bank 1 Busy**[1)]
P0BUSY	3	rh	**Program Flash PF0 Busy**[1)] HW-controlled status flag. 0_B PF0 ready, not busy; PF0 in read mode. 1_B PF0 busy; PF0 not in read mode. Indication of busy state of PF0 because of active execution of an operation; PF0 busy state is also indicated during Flash startup after reset or in sleep mode; while in busy state, the PF0 is not in read mode.
RES4	4	rh	**Reserved for Program Flash PF1 Busy**[1)]
RES5	5	rh	**Reserved for Program Flash PF2 Busy**[1)]
RES6	6	rh	**Reserved for Program Flash PF3 Busy**[1)]

Field	Bits	Type	Description
PROG	7	rwh	**Programming State** [2)3)] HW-controlled status flag. 0_B There is no program operation requested or in progress or just finished. 1_B Programming operation requested or in action or finished. Set with last cycle of Write Page/Burst command sequence, cleared with Clear Status command (if not busy) or with power-on reset. If one BUSY flag is coincidently set, PROG indicates the type of busy state. If OPER is coincidently set, PROG indicates the type of erroneous operation. Otherwise, PROG indicates, that operation is still requested or finished. Can be also cleared by writing '1' to it.
ERASE	8	rwh	**Erase State** [2)3)] HW-controlled status flag. 0_B There is no erase operation requested or in progress or just finished 1_B Erase/Verify operation requested or in action or finished. Set with last cycle of Erase/Verify command sequence, cleared with Clear Status command (if not busy) or with power-on reset. Indications are analogous to PROG flag. Can be also cleared by writing '1' to it.
PFPAGE	9	rh	**Program Flash in Page Mode**[1)4)] HW-controlled status flag. 0_B Program Flash not in page mode. 1_B Program Flash in page mode. Set with Enter Page Mode for PFlash, cleared with Write Page command *Note: Concurrent page and read modes are allowed*
DFPAGE	10	rh	**Data Flash in Page Mode**[1)4)] HW-controlled status flag. 0_B Data Flash not in page mode 1_B Data Flash in page mode Set with Enter Page Mode for DFlash, cleared with Write Page command. *Note: Concurrent page and read modes are allowed*

Field	Bits	Type	Description
OPER	11	rwh	**Flash Operation Error[2)3)4)]** 0_B No operation error. 1_B Flash array operation aborted, because of a Flash array failure, e.g. an ECC error in microcode. This bit is not cleared with application reset, but with power-on reset. Registered status bit; must be cleared per command or by writing ‘1’.
SQER	12	rwh	**Command Sequence Error[1)2)4)]** 0_B No sequence error 1_B Command state machine operation unsuccessful because of improper address or command sequence. A sequence error is not indicated if the Reset to Read command aborts a command sequence. Registered status bit; must be cleared per command or by writing ‘1’.
PROER	13	rwh	**Protection Error[1)2)4)]** 0_B No protection error 1_B Protection error. A Protection Error is reported e.g. because of a not allowed command, for example an Erase or Write Page command addressing a locked sector, or because of wrong password(s) in a protected command sequence such as “Disable Read Protection”. **A Protection Error is also reported if the safety protection (Chapter 11.5.5.2) prevented a program/erase operation in PFlash.** Registered status bit; must be cleared per command or by writing ‘1’.
PFSBER	14	rwh	**PFlash Single-Bit Error and Correction[1)2)4)]** 0_B No Single-Bit Error detected during read access to PFlash. 1_B Single-Bit Error detected. Registered status bit; must be cleared per command or by writing ‘1’.

Field	Bits	Type	Description
PFDBER	15	rwh	**PFlash Double-Bit Error[1)2)4)]** 0_B No Double-Bit Error detected during read access to PFlash. 1_B Double-Bit Error detected in PFlash. Registered status bit; must be cleared per command or by writing ‘1’.
PFMBER	16	rwh	**PFlash Uncorrectable Error[1)2)4)]** 0_B No uncorrectable error (3-bit error or more, address error, all-0/all-1) detected during read access to PFlash. 1_B Uncorrectable error (3-bit error or more, address error, all-0/all-1) detected in PFlash. Registered status bit; must be cleared per command or by writing ‘1’.
RES17	17	rwh	**Reserved[1)2)4)]** Registered status bit; must be cleared per command or by writing ‘1’.
DFSBER	18	rwh	**DFlash Single-Bit Error[1)2)4)]** 0_B No Single-Bit Error detected during read access to DFlash. 1_B Single-Bit Error detected. Registered status bit; must be cleared per command or by writing ‘1’.
DFDBER	19	rwh	**DFlash Double-Bit Error[1)2)4)]** 0_B No Double-Bit Error detected during read access to DFlash. 1_B Double-Bit Error detected. Registered status bit; must be cleared per command or by writing ‘1’.
DFTBER	20	rwh	**DFlash Triple-Bit Error[1)2)4)]** 0_B No Triple-Bit Error detected during read access to DFlash. 1_B Triple-Bit Error detected. Registered status bit; must be cleared per command or by writing ‘1’.

Field	Bits	Type	Description
DFMBER	21	rwh	**DFlash Uncorrectable Error[1)2)4)]** 0_B No uncorrectable error detected during read access to DFlash 1_B Uncorrectable error detected in DFlash. Registered status bit; must be cleared per command or by writing ‘1’.
SRIADDERR	22	rwh	**SRI Bus Address ECC Error[1)2)4)]** This flag is set when the PMU detects an ECC error in the address phase bus transaction on the SRI bus. 0_B No SRI address error detected. 1_B SRI address error detected. Registered status bit; must be cleared by writing ‘1’.
RES	24	rwh	**Reserved** Write zero. Read value unpredictable.
PVER	25	rwh	**Program Verify Error[1)2)4)]** A verify error was reported during a Flash program operation. 0_B The page is correctly programmed. All bits have full expected quality. 1_B A program verify error has been detected. Full quality of all bits cannot be guaranteed. Registered status bit; must be cleared per command or writing ‘1’.
EVER	26	rwh	**Erase Verify Error[1)2)4)]** A verify error was reported during a Flash erase operation. 0_B The sector is correctly erased. All erased bits have full expected quality. 1_B An erase verify error has been detected. Full quality erased bits cannot be guaranteed. Registered status bit; must be cleared per command or writing ‘1’.

Field	Bits	Type	Description
SPND	27	rwh	**Operation Suspended**[3)] Requested by MARD.SPND a program or erase operation is suspended. 0_B No Flash operation is suspended. 1_B Suspended Flash operation. Can be also cleared by writing '1' to it to clean up after a reset that killed a suspended operation context.
SLM	28	rh	**Flash Sleep Mode**[1)] HW-controlled status flag. Indication of Flash sleep mode taken because of global or individual sleep request; additionally indicates when the Flash is in shut down mode. 0_B Flash not in sleep mode 1_B Flash is in sleep or shut down mode
ORIER	30	rh	**Original Error**[1)2)4)] 0_B No original error detected during startup. 1_B Original data replaced by its copy.
0	31, 29, 23	r	**Reserved** Read zero, no write

1) Cleared with application reset
2) Cleared with command "Clear Status"
3) Cleared with power-on reset (PORST)
4) Cleared with command "Reset to Read"

Note: The xBUSY flags cannot be cleared with the "Clear Status" command or with the "Reset to Read" command. These flags are controlled by HW.

11.7.2.4 Flash Configuration Control

The Flash Configuration Register FCON reflects and controls the general Flash configuration functions.

FCON
Flash Configuration Register (1014_H) Reset value: 0091 XXXX$_H$[1)]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EOBM	PR5V	0			PROERM	SQERM	VOPERM	RES23		RES21		STALL	NSAFEC	SLEEP	ESLDIS
rw	rw	r			rw	rw	rw	rh		rh		rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDLE	WSECDF			WSDFLASH						WSECPF		WSPFLASH			
rw	rw			rw						rw		rw			

1) The wait-cycles WSECDF, WSDFLASH, WSECPF and WSPFLASH are changed by the startup after system and power-on resets. **Attention: the configured value is only sufficient for the clock configuration used during startup.** The wait-cycles have to be configured after startup as described in **Chapter 11.5.3.3** before changing to higher clock frequencies.

Field	Bits	Type	Description
WSPFLASH	[3:0]	rw	**Wait States for read access to PFlash** This bitfield defines the number of wait states in number of FSI2 clock cycles, which are used for an initial read access to the Program Flash memory area (see **Chapter 11.5.3.3**). 0_D Read access takes 1 FSI2 clock cycle. 1_D Read access takes 2 FSI2 clock cycles. $...{}_D$ ...
WSECPF	[5:4]	rw	**Wait States for Error Correction of PFlash** This bitfield defines the number of wait-states for the ECC correction in FSI2 clock cycles (see **Chapter 11.5.3.3**). 0_D Error correction takes 1 FSI2 clock cycle. 1_D Error correction takes 2 FSI2 clock cycles. $...{}_D$ ...

Field	Bits	Type	Description
WSDFLASH	[11:6]	rw	**Wait States for read access to DFlash** This bitfield defines the number of wait states in number of FSI clock cycles, which are used for a read access to the Data Flash memory area (see **Chapter 11.5.3.3**). 0_D Read access takes 1 FSI clock cycle. 1_D Read access takes 2 FSI clock cycles. $...{}_D$ ...
WSECDF	[14:12]	rw	**Wait State for Error Correction of DFlash** This bitfield defines the number of wait-states for the ECC correction in FSI clock cycles (see **Chapter 11.5.3.3**). 0_D Error correction takes 1 FSI clock cycle. 1_D Error correction takes 2 FSI clock cycles. $...{}_D$ ...
IDLE	15	rw	**Dynamic Flash Idle** 0_B Normal/standard Flash read operation 1_B Dynamic idle of Program Flash enabled for power saving; static prefetching disabled *Note: In Data Flash, dynamic idle is always enabled (prefetching not supported).* Initial value after startup is 0_B.
ESLDIS	16	rw	**External Sleep Request Disable** 0_B External sleep request signal input is enabled 1_B Externally requested Flash sleep is disabled The ‘external’ signal input is connected with a global power-down/sleep request signal from SCU.
SLEEP	17	rw	**Flash SLEEP** 0_B Normal state or wake-up 1_B Flash sleep mode is requested, Wake-up from sleep is started with clearing of the SLEEP-bit.

Field	Bits	Type	Description
NSAFECC	18	rw	**Non-Safety PFlash ECC** This bit selects if the data in PFlash is written and read with the “Safety” ECC code (calculated over address bits), or the “Legacy” ECC code (see **Chapter 11.5.6**). 0_B Safety ECC is used. 1_B Legacy ECC is used. **Attention: this bit must not be changed while accessing Flash (reading, programming, erasing).** Initial value after startup depends on PROCOND.NSAFECC.
STALL	19	rw	**Stall SRI** This field selects if reading from busy Flash banks causes a bus error or wait-states until busy is cleared again. 1_B **Stall**, Reading busy Flash banks suppresses the SRI bus ready effectively causing wait-states until busy is cleared. 0_B **Error**, Reading busy Flash banks causes a bus error. *Note: This field must not be changed while any bank is busy. The results are unpredictable. Generally it’s strongly recommended to configure this field once and avoid changing it during operation.* *Note: Reading Flash in sleep mode causes always a bus error independent of this field (although it reports “busy”).*
RES21	[21:20]	rh	**Reserved** Write 0; Read value unpredictable.
RES23	[23:22]	rh	**Reserved** Write 0; Read value unpredictable.
VOPERM	24	rw	**Verify and Operation Error Interrupt Mask** 0_B Interrupt not enabled 1_B Flash interrupt because of Verify Error or Operation Error in Flash array (FSI) is enabled
SQERM	25	rw	**Command Sequence Error Interrupt Mask** 0_B Interrupt not enabled 1_B Flash interrupt because of Sequence Error is enabled

Field	Bits	Type	Description
PROERM	26	rw	**Protection Error Interrupt Mask** 0_B Interrupt not enabled 1_B Flash interrupt because of Protection Error is enabled
EOBM	31	rw	**End of Busy Interrupt Mask** 0_B Interrupt disabled. 1_B EOB interrupt is enabled.
PR5V	30	rw	**Programming Supply 5V** Selects the supply for programming. **Attention: In this 3.3 V device the setting "P5V" is forbidden!** 0_B **P3V**, The programming voltage is internally generated. 1_B **P5V**, As programming voltage the external 5 V supply is used. **Forbidden!**
0	[29:27]	r	**Reserved** Write 0; Read 0.

Note: The default numbers of wait states represent the slow cases. This is a general proceeding and additionally opens the possibility to execute higher frequencies without changing the configuration.

11.7.2.5 Flash Protection

Flash Protection Control and Status Register

This register reports the state of the Flash protection (see **Chapter 11.5.5.5**) and contains protection relevant control fields.

FPRO
Flash Protection Control and Status Register
($101C_H$) **Reset Value: $00XX\ 00A0_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								ENPE		0	DDFD	0	DDFPX	DDFP	DCFP
r								rw		r	rwh	r	rwh	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					PROINHSM	PRODISDBG	PROINDBG	RES7	PROINOTP	RES5	PROINHSMCOTP	PRODISD	PROIND	PRODISP	PROINP
r					rh	rh	rh	r	rh	r	rh	rh	rh	rh	rh

Field	Bits	Type	Description
PROINP	0	rh	**PFlash Protection** This bit reflects the confirmed state of UCB_PFlash.
PRODISP	1	rh	**PFlash Protection Disabled**[1)] The protection configured by UCB_PFlash was successfully disabled by supplying the correct password to “Disable Protection”.
PROIND	2	rh	**DFlash Protection** This bit reflects the confirmed state of UCB_DFlash.
PRODISD	3	rh	**DFlash Protection Disabled**[1)] The protection configured by UCB_DFlash was successfully disabled by supplying the correct password to “Disable Protection”.
PROINHSMCOTP	4	rh	**HSM OTP Protection** This bit reflects the confirmed state of UCB_HSMCOTP.
RES5	5	rh	**Reserved** Read value don’t care, updated during startup; should be written with 0.
PROINOTP	6	rh	**OTP and Write-Once Protection** This bit reflects the confirmed state of UCB_OTP.

Field	Bits	Type	Description
RES7	7	rh	**Reserved** Read value don't care, updated during startup; should be written with 0.
PROINDBG	8	rh	**Debug Interface Password Protection** This bit reflects the confirmed state of UCB_DBG.
PRODISDBG	9	rh	**Debug Interface Password Protection Disabled**[1)] The password configured by UCB_DBG was correctly received with "Disable Protection". When PROCONHSMCOTP.DESTDBG is "destructive" then only the SSW can disable this protection.
PROINHSM	10	rh	**HSM Configuration** This bit reflects the confirmed state of UCB_HSM.
DCFP	16	rwh	**Disable Code Fetch from PFlash Memory for CPU0 PMI** This bit enables/disables the code fetch from the internal PFlash memory area for CPU0 PMI (see **Chapter 11.5.5.3**). Once set, this bit can only be cleared when FPRO.PRODISP or not PROCONP0.RPRO. This bit is automatically set with reset and is cleared during startup, if no RP installed, and during startup (BootROM SW) in case of internal start out of Flash. 0_B Code fetching by CPU0 from the PFlash memory area is allowed. 1_B Code fetching by CPU0 from the PFlash memory area is not allowed.

Field	Bits	Type	Description
DDFP	17	rwh	**Disable Read from PFlash for CPU0 DMI** This bit enables/disables the read access to the PFlash memory area by for CPU0 DMI (see **Chapter 11.5.5.3**). Once set, this bit can only be cleared when FPRO.PRODISP or not PROCONP0.RPRO. This bit is automatically set with reset and is cleared during startup, if no RP installed, and during startup (BootROM SW) in case of internal start out of Flash. 0_B Read access to the PFlash memory area is allowed. 1_B Read access to the PFlash memory area is not allowed.
DDFPX	18	rwh	**Disable Read from PFlash for Other Masters** This bit enables/disables the read access to PFlash for other masters (see **Chapter 11.5.5.3**). Once set, this bit can only be cleared when FPRO.PRODISP or not PROCONP0.RPRO. This bit is automatically set with reset and is cleared during startup, if no RP installed, and during startup (BootROM SW) in case of internal start out of Flash. 0_B The read access by other masters to the PFlash memory area is allowed. 1_B The read access to the PFlash memory area is not allowed for other masters.
DDFD	20	rwh	**Disable Data Fetch from DFlash Memory** This bit enables/disables the data fetch from the internal DFlash memory area (see **Chapter 11.5.5.3**). Once set, this bit can only be cleared when FPRO.PRODISD or not PROCOND.RPRO. This bit is automatically set with reset and is cleared during startup, if no RP installed, and during startup (BootROM SW) in case of internal start out of Flash. 0_B Data fetching from the DFlash memory area is allowed. 1_B Data fetching from the DFlash memory area is not allowed.

Field	Bits	Type	Description
ENPE	[23:22]	rw	**Enable Program/Erase** The field prevents any Flash program or erase directly in the FSI. It is set to Enabled by the SSW upon finishing the startup. 01_B **Enabled**, Program/Erase are enabled in the FSI. 10_B **Disabled**, Program/Erase are disabled in the FSI. others: Illegal values, disabled also Program/Erase. Once in “Enabled” state this field can’t be changed anymore.
0	[31:24], 21, 19, [15:11]	r	**Reserved** Always read as 0; should be written with 0.

1) Cleared with command “Resume Protection”.

Notes

After reset and execution of BootROM startup SW, the read protection control bits are set to the following values:

- Start not in internal Flash:
  - DCFP set to PROCONP0.RPRO.
  - DDFP set to PROCONP0.RPRO.
  - DDFPX set to PROCONP0.RPRO.
  - DDFD set to PROCOND.RPRO.
- Start in internal Flash:
  - DCFP, DDFP, DDFPX, DDFD set to 0.
- Independent of the start configuration: the PROIN fields are set depending on the content of their assigned UCB sectors.

Attention! Before disabling read access by setting any of the DCF* and DDF* flags all pending read accesses to the affected Flash ranges should have finished.

11.7.2.6 Protection Configuration

The configuration of read/write/OTP protection is indicated with registers PROCONP, PROCOND, PROCONHSMCOTP, PROCONOTP, PROCONWOP, PROCONHSM and PROCONDBG.

PFlash Protection Configuration

When UCB_PFlash is in errored state the initial value is “FFFF FFFF$_H$”, else the initial value after Flash startup reflects the UCB_PFlash content (**Table 11-8**).

PROCONPp (p=0-0)
PFlash Protection Configuration PFp

(1020$_H$+p*4$_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RPRO	RES				S26L	S25L	S24L	S23L	S22L	S21L	S20L	S19L	S18L	S17L	S16L
rh	rh				rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S15L	S14L	S13L	S12L	S11L	S10L	S9L	S8L	S7L	S6L	S5L	S4L	S3L	S2L	S1L	S0L
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

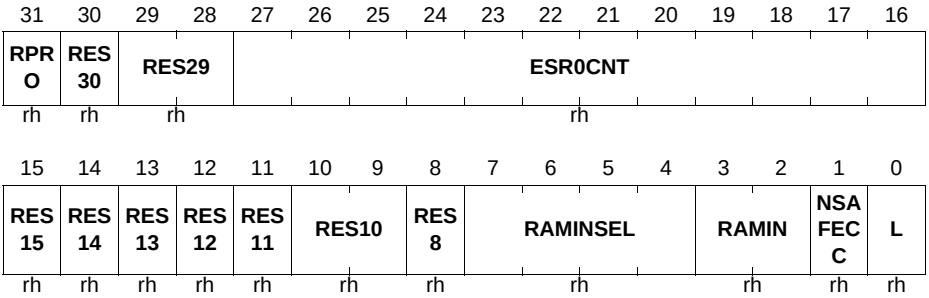
Field	Bits	Type	Description
SxL (x=0-26)	x	rh	**PFlash p Sector x Locked for Write Protection** These bits indicate whether PFLASH sector x is write-protected. 0_B No write protection is configured for sector x. 1_B Write protection is configured for sector x. In PMU0 PROCONP0 the S6L, S16L and S17L are only effective if these HSM code sectors are not HSM_exclusive.
RPRO	31	rh	**Read Protection Configuration** This bit indicates whether read protection is configured for PFLASH. This bit is only used in PROCONP0. In other PROCONPs it is reserved. 0_B No read protection configured 1_B Read protection and global write protection is configured.
RES	[30:27]	rh	**Reserved** Deliver the corresponding content of UCB_PFlash.

DFlash Protection Configuration

Only existing in PMUs with DFlash.

When UCB_DFlash is in errored state the initial value is “8000 0000$_H$”, else the initial value after Flash startup reflects the UCB_DFlash content (**Table 11-9**).

PROCOND
DFlash Protection Configuration (1030$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
L	0	rh	**DF_EEPROM Locked for Write Protection** This bit indicates whether the DFlash sectors EEPROMx are write protected. 0_B No write protection is configured. 1_B Write protection is configured.
NSAFECC	1	rh	**Non-Safety PFlash ECC** This bit indicates whether the SSW selects the “Safety” PFlash ECC algorithm or the “Legacy” ECC. 0_B Safety ECC is configured. 1_B Legacy ECC is configured.

Field	Bits	Type	Description
RAMIN	[3:2]	rh	**RAM Initialization by SSW Control** These bits defined whether the RAMs selected by the field RAMINSEL are initialized. 00_B **Init_All**, RAM initialization is performed after cold power-on resets and warm power-on resets. 01_B **Init_Warm**, RAM initialization is performed after warm power-on resets but not after cold power-on resets (not recommended). 10_B **Init_Cold**, RAM initialization is performed after cold power-on resets. 11_B **No_Init**, No RAM initialization is performed. This field determines also if a RAM is initialized before MBIST access is granted. In all “Init_*” cases a RAM is initialized before MBIST access is enabled, in the “No_Init” case a RAM is not erased. This is independent of the memory selection with RAMINSEL for initialization during startup (see MTU chapter).
RAMINSEL	[7:4]	rh	**RAM Initialization Selection** These bits select which memories are initialized when the RAM initialization is configured with RAMIN. Each bit with value ‘0’ selects the corresponding memory for initialization. $xxx0_B$ RAMs (PSPR, DSPR, PCACHE and DCACHE) in CPU0 are selected for initialization. $xx0x_B$ Reserved for CPU1. $x0xx_B$ Reserved for CPU2. $0xxx_B$ LMU RAMs are selected for initialization.
RESx(x=8-15)	x	rh	**Reserved** Deliver the corresponding content of UCB_DFlash.
ESR0CNT	[27:16]	rh	**ESR0 Prolongation Counter** Used to configure the ESR0 delay. Evaluation by SSW.
RES29	[29:28]	rh	**Reserved** Deliver the corresponding content of UCB_DFlash.

Field	Bits	Type	Description
RES30	30	rh	**Reserved** Deliver the corresponding content of UCB_DFlash.
RPRO	31	rh	**Read Protection Configuration** This bit indicates whether read protection is configured for DFlash sectors EEPROMx. 0_B No read protection configured 1_B Read protection and write protection is configured.

OTP Protection Configuration

After Flash startup this register represents the or-combination of all PROCONOTP entries of all confirmed configuration sets in UCB_OTP (**Table 11-10**).

When any OTP set is in the errored state the initial content is forced to "3FFF FFFF$_H$".

PROCONOTPp (p=0-0)
OTP Protection Configuration PFp

(1038_H+p*4_H) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TP	**BML**		**RES**		**S26 ROM**	**S25 ROM**	**S24 ROM**	**S23 ROM**	**S22 ROM**	**S21 ROM**	**S20 ROM**	**S19 ROM**	**S18 ROM**	**S17 ROM**	**S16 ROM**
rh	rh		rh		rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S15 ROM	**S14 ROM**	**S13 ROM**	**S12 ROM**	**S11 ROM**	**S10 ROM**	**S9R OM**	**S8R OM**	**S7R OM**	**S6R OM**	**S5R OM**	**S4R OM**	**S3R OM**	**S2R OM**	**S1R OM**	**S0R OM**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
SxROM (x=0-26)	x	rh	**PFlash p Sector x Locked Forever** These bits indicate whether PFlash p sector x is an OTP protected sector with read-only functionality. 0_B No OTP protection is configured for sector x. 1_B OTP protection is configured for sector x. In PMU0 PROCONOTP0 the S6ROM, S16ROM and S17ROM are not effective. These HSM code sectors are OTP protected by PROCONHSMCOTP.

Field	Bits	Type	Description
TP	31	rh	**Tuning Protection** This bit indicates whether tuning protection is installed or not. This bit is only evaluated in PMU0 PROCONOTP0. In other PMUs and PROCONOTPs it is reserved. 0_B Tuning protection is not configured. 1_B Tuning protection is configured and installed, if correctly confirmed.
RES	[28:27]	rh	**Reserved** Deliver the corresponding content of UCB_OTP.
BML	[30:29]	rh	**Boot Mode Lock** Only used in PMU0 PROCONOTP0, in other PROCONOTPs these bits are reserved. Used by the SSW to restrict the boot mode selection. 00_B Boot flow with standard evaluation of boot headers. 01_B Restricted boot flow, never evaluating HWCFG pins and without fallback to boot loader. 10_B Restricted boot flow, never evaluating HWCFG pins and without fallback to boot loader. 11_B Restricted boot flow, never evaluating HWCFG pins and without fallback to boot loader.

Write-Once Protection Configuration

After Flash startup this register represents the or-combination of all PROCONWOP entries of all confirmed configuration sets in UCB_OTP (**Table 11-10**).

When any OTP set is in the errored state the initial content is forced to “FFFF FFFF$_H$”.

PROCONWOPp (p=0-0)
Write-Once Protection Configuration PFp

(1048_H+p*4_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATM	RES				S26WOP	S25WOP	S24WOP	S23WOP	S22WOP	S21WOP	S20WOP	S19WOP	S18WOP	S17WOP	S16WOP
rh	rh				rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S15WOP	S14WOP	S13WOP	S12WOP	S11WOP	S10WOP	S9WOP	S8WOP	S7WOP	S6WOP	S5WOP	S4WOP	S3WOP	S2WOP	S1WOP	S0WOP
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
SxWOP (x=0-26)	x	rh	**PFlash p Sector x Configured for Write-Once Protection** These bits indicate whether PFlash p sector x is an WOP protected sector. 0_B No WOP protection is configured for sector x. 1_B WOP protection is configured for sector x. In PMU0 PROCONWOP0 the S6WOP, S16WOP and S17WOP are not effective. These HSM code sectors are OTP protected by PROCONHSMCOTP.
DATM	31	rh	**Disable ATM** This bit indicates if the ATM “Application Test Mode” is disabled or not. 0_B ATM is enabled. 1_B ATM is disabled. This bit is only used in the PMU0 PROCONWOP0. In other PMUs and PROCONWOPs it is reserved.
RES	[30:27]	rh	**Reserved** Deliver the corresponding content of UCB_OTP.

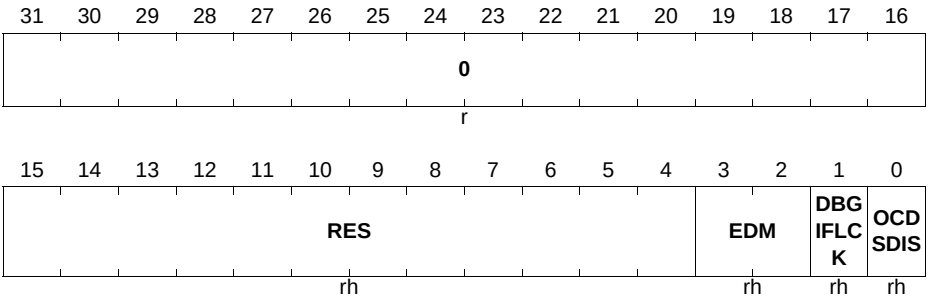
Debug Interface Protection Configuration

The debug interface and OCDS enable have a dedicated password protection logic.

When UCB_DBG is in errored state the initial value is “0000 $FFFF_H$”, else the initial value after Flash startup reflects the UCB_DBG content (**Table 11-12**).

PROCONDBG
Debug Interface Protection Configuration

(1058_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
OCDSDIS	0	rh	**OCDS Disabled** This bit indicates whether the OCDS is configured as locked. 0_B No OCDS lock configured in UCB_DBG. 1_B OCDS lock configured in UCB_DBG.
DBGIFLCK	1	rh	**Debug Interface Locked** This bit indicates whether the debug interface is configured as locked. 0_B No debug interface lock configured in UCB_DBG. 1_B Debug interface lock configured in UCB_DBG.

Field	Bits	Type	Description
EDM	[3:2]	rh	**Entered Debug Mode** This bit indicates whether the debug interface has been opened via the "**Destructive Debug Entry**" (**Page 11-37**). Consequently the CAN and Flexray operation is made impossible! 00_B "Debug not entered", device operation not affected. 01_B "Debug not entered", device operation not affected. 10_B "Debug not entered", device operation not affected. 11_B "Debug entered", CAN and Flexray operation affected.
RES	[15:4]	rh	**Reserved** Deliver the corresponding content of UCB_DBG.
0	[31:16]	r	**Reserved** Always read as 0; should be written with 0.

HSM Code Flash OTP Protection Configuration

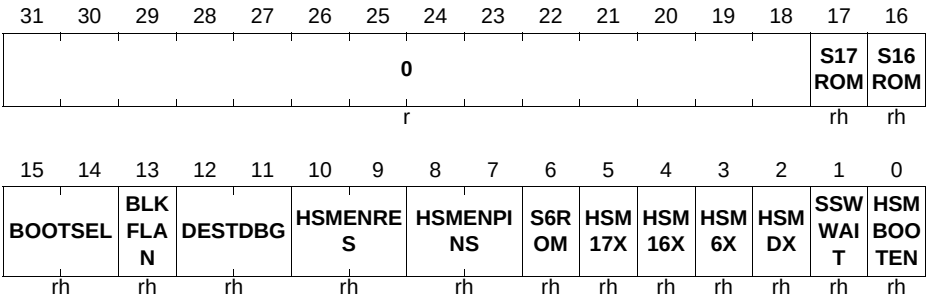
The HSM code sectors have also their own OTP protection configuration. This register represents after Flash startup the or-combination of all PROCONHSMCOTP entries of all confirmed configuration sets in UCB_HSMCOTP (see **Table 11-13**).

When any OTP set is in the errored state the initial content is forced to "0003 $007F_H$".

When configuring any HSM_exclusive protection it is strongly recommended to activate all four HSM_exclusive bits.

When enabling HSM boot with HSMBOOTEN is it strongly recommended to use all HSM code sectors only for HSM code but not for Tricore code or data.

PROCONHSMCOTP
HSM Code Flash OTP Protection Configuration
(1034_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
HSMBOOTEN	0	rh	**HSM Boot Enable** 0_B HSM Boot is not enabled. 1_B HSM Boot is enabled. HSMBOOTEN must not be set to 1 in devices without HSM.
SSWWAIT	1	rh	**SSW Wait** Defines if the SSW waits for the HSM to release the jump of CPU0 to user code. 0_B The SSW does not wait for HSM. 1_B SSW wait for acknowledge of HSM.
HSMDX	2	rh	**HSM Data Sectors Exclusive** This bit indicates whether the HSM data sectors HSMx are configured as “HSM_exclusive”. 0_B HSMx are not HSM_exclusive. 1_B HSMx are HSM_exclusive.
HSM6X	3	rh	**HSM Code Sector 6 Exclusive** This bit indicates whether the HSM code sector S6 is configured as “HSM_exclusive”. 0_B S6 is not HSM_exclusive. 1_B S6 is HSM_exclusive.

Field	Bits	Type	Description
HSM16X	4	rh	**HSM Code Sector 16 Exclusive** This bit indicates whether the HSM code sector S16 is configured as “HSM_exclusive”. 0_B S16 is not HSM_exclusive. 1_B S16 is HSM_exclusive.
HSM17X	5	rh	**HSM Code Sector 17 Exclusive** This bit indicates whether the HSM code sector S17 is configured as “HSM_exclusive”. 0_B S17 is not HSM_exclusive. 1_B S17 is HSM_exclusive.
S6ROM	6	rh	**HSM Code Sector 6 Locked Forever** This bit indicates whether PFlash sector 6 is an OTP protected sector with read-only functionality. 0_B No OTP protection is configured. 1_B OTP protection is configured.
HSMENPINS	[8:7]	rh	**Enable HSM Forcing of Pins HSM1/2** This bit indicates whether HSM may force the value of the pins HSM1/2 (i.e. overrule the value driven by the application). 00_B HSM can’t force pins. 01_B HSM can’t force pins. 10_B HSM can’t force pins. 11_B HSM can force pins.
HSMENRES	[10:9]	rh	**Enable HSM Triggering Resets** This bit indicates whether HSM may trigger application or system resets. 00_B HSM can’t trigger resets. 01_B HSM can’t trigger resets. 10_B HSM can’t trigger resets. 11_B HSM can trigger resets.
DESTDBG	[12:11]	rh	**Destructive Debug Entry** This field configures the destructive debug entry (see **Destructive Debug Entry** on **Page 11-37**). 00_B Debug entry is non-destructive. 01_B Debug entry is non-destructive. 10_B Debug entry is non-destructive. 11_B Debug entry is destructive.

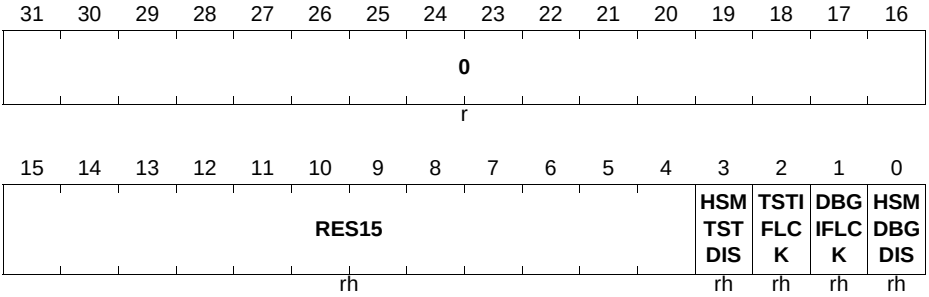
Field	Bits	Type	Description
BLKFLAN	13	rh	**Block Flash Analysis** This bit allows to block the command “Verify Erased Logical Sector Range” on certain HSM related Flash ranges as described on **Page 11-21**. 0_B Functions allowed on all Flash ranges. 1_B Functions blocked on HSM_exclusive Flash ranges.
BOOTSEL	[15:14]	rh	**Boot Sector Selection** This field controls which of the HSM code sectors S6, S16, S17 is searched for boot code. 00_B Sectors S6, S16 and S17 are searched. 01_B Sectors S6 and S16 are searched. 10_B Sector S6 is searched. 11_B Sector S17 is searched.
S16ROM	16	rh	**HSM Code Sector 16 Locked Forever** This bit indicates whether PFlash sector 16 is an OTP protected sector with read-only functionality. 0_B No OTP protection is configured. 1_B OTP protection is configured.
S17ROM	17	rh	**HSM Code Sector 17 Locked Forever** This bit indicates whether PFlash sector 17 is an OTP protected sector with read-only functionality. 0_B No OTP protection is configured. 1_B OTP protection is configured.
0	[31:18]	r	**Reserved** Always read as 0; should be written with 0.

HSM Interface Configuration

Only existing in PMU0. Content only evaluated in devices with activated HSM.

When UCB_HSM is in errored state the initial value is “0000 FFFF$_H$”, else the initial value after Flash startup reflects the UCB_HSM content (**Table 11-14**).

PROCONHSM
HSM Interface Configuration (105C$_H$) **Reset Value: 0000 0000$_H$**



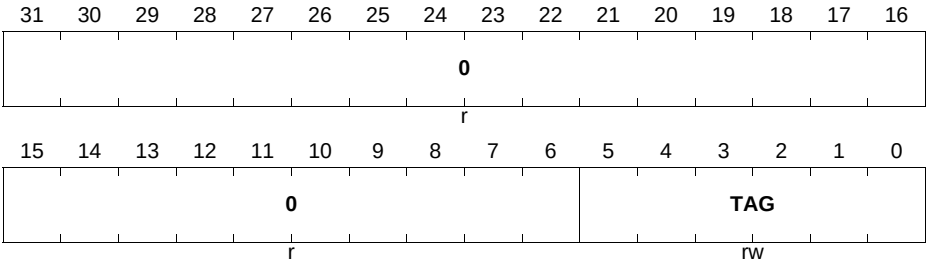
Field	Bits	Type	Description
HSMDBGDIS	0	rh	**HSM Debug Disable** This bit indicates whether HSM debug is configured as “disabled”. 0_B HSM debug is enabled. 1_B HSM debug is disabled.
DBGIFLCK	1	rh	**Debug Interface Locked** This bit indicates whether the chip debug interface is configured as “locked”. 0_B Debug is unlocked. 1_B Debug is locked.
TSTIFLCK	2	rh	**Test Interface Locked** This bit indicates whether the chip test interface is configured as “locked”. 0_B Test interface is unlocked. 1_B Test interface is locked.
HSMTSTDIS	3	rh	**HSM Test Disable** This bit indicates whether the HSM test is configured as “disabled”. 0_B HSM test is enabled. 1_B HSM test is disabled.
RES15	[15:4]	rh	**Reserved** Deliver the corresponding content of UCB_HSM.
0	[31:16]	r	**Reserved** Always read as 0; should be written with 0.

11.7.2.7 Flash Read Buffer Configuration

The RDBCFGp0–2 registers define the assignment of read buffers to masters for each PFlash p read port. The functionality of the read buffers is described in **Chapter 11.5.3**.

Assigning the same master tag to more than one of the 3 read buffers of one read port does not increase the performance further.

RDBCFGp0 (p=0-0)
Read Buffer Port p Cfg 0 **(1060_H+p*C_H)** **Reset Value: 0000 0000_H**
RDBCFGp1 (p=0-0)
Read Buffer Port p Cfg 1 **(1064_H+p*C_H)** **Reset Value: 0000 0002_H**
RDBCFGp2 (p=0-0)
Read Buffer Port p Cfg 2 **(1068_H+p*C_H)** **Reset Value: 0000 0004_H**



Field	Bits	Type	Description
TAG	[5:0]	rw	**Master Tag** This read buffer is assigned to the master with SRI tag = TAG.
0	[31:6]	r	**Reserved** Write 0, read 0.

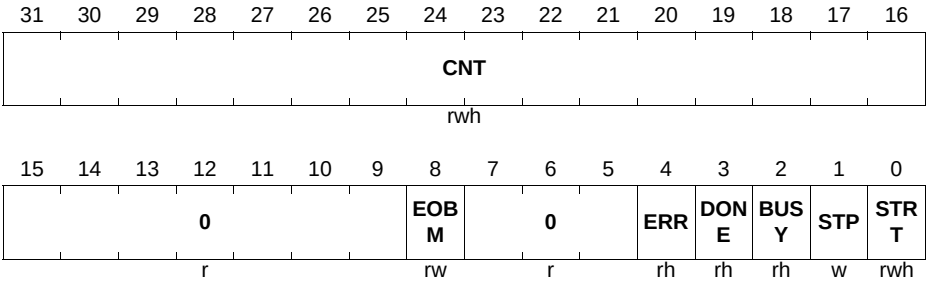
11.7.2.8 Requested Read Interface

The following registers can be used by any CPU to control the requested read interface and perform data transfers (see **Chapter 11.5.3.4**) from the EEPROMx sectors.

Requested Read Control

RRCT
Requested Read Control Register

(1140$_H$) Reset Value: 0000 0000$_H$



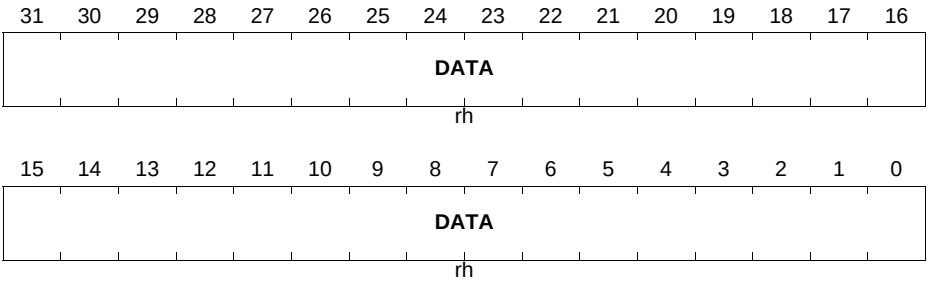
Field	Bits	Type	Description
STRT	0	rwh	**Start Request** Writing ‘1’ to this bit starts the next read process as configured by RRAD and RRCT.CNT. This bit is cleared by the PMU as soon the reading process starts executing.
STP	1	w	**Stop** Stops the read process.
BUSY	2	rh	**Flash Read Busy** 0_B The RRD registers contain the data addressed by RRAD (data ready). 1_B The Flash is busy reading the data addressed by RRAD.
DONE	3	rh	**Flash Read Done** 0_B The Flash read has not finished. 1_B The Flash read has finished, data is available.
ERR	4	rh	**Error** Cleared when starting a sequence with STRT, set when an error is detected. 0_B No error occurred. 1_B Error occurred.

Field	Bits	Type	Description
EOBM	8	rw	**End of Busy Interrupt Mask** Interrupt for RRCT.BUSY. 0_B Interrupt disabled. 1_B EOB interrupt is enabled.
CNT	[31:16]	rwh	**Count** Defines number of remaining data transfers.
0	[15:9], [7:5]	r	**Reserved** Always read as 0; should be written with 0.

Requested Read Data Registers

RRDx (x=0-1)
Requested Read Data Register x

(1144_H+x*4_H) **Reset Value: 0000 0000_H**

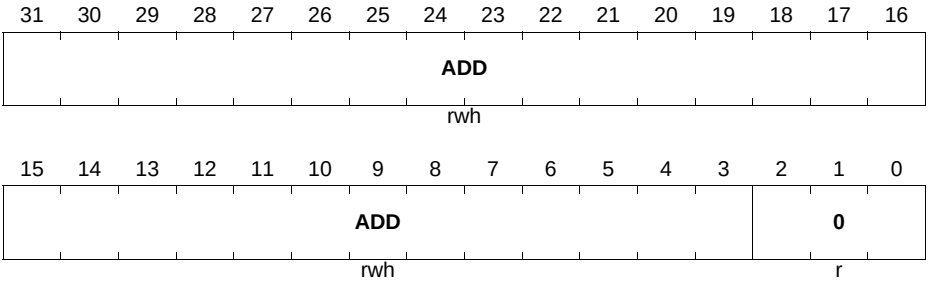


Field	Bits	Type	Description
DATA	[31:0]	rh	**Read Data** Read data.

Requested Read Address Register

RRAD
Requested Read Address Register

(114C$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
ADD	[31:3]	rwh	**Address** Start address/current address Can be written by software to define the next read address. Is automatically incremented by the requested read hardware when finishing the last read access. In case both writes happen concurrently the hardware update is performed, the software write is ignored.
0	[2:0]	r	**Reserved** Always read as 0; should be written with 0.

11.7.2.9 Flash ECC Access

ECC Write Register

The Error Correction Code Write register ECCW contains bits for disabling the ECC encoding separately for PFlash and DFlash.

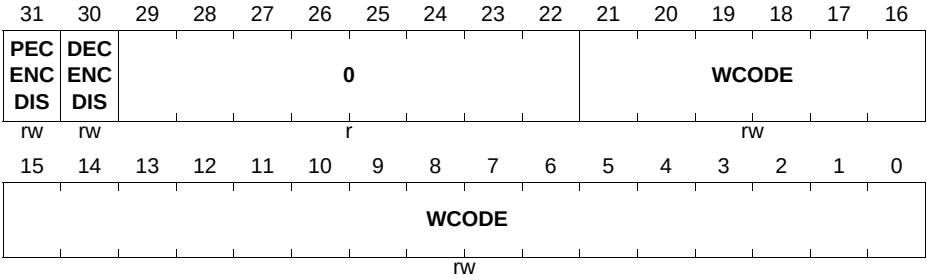
When disabling the ECC encoding for PFlash with PECENCDIS = ‘1’ the ECC code for the next 256-bit data block transferred from PMU to the Flash assembly buffer is taken from ECCW.WCODE.

When disabling the ECC encoding for DF_EEPROM with DECENCDIS = ‘1’ the ECC code for the next 64-bit data block transferred from PMU to the Flash assembly buffer is taken from ECCW.WCODE.

Because of internal dependencies only one of PECENCDIS or DECENCDIS may be set to ‘1’.

When any of PECENCDIS or DECENCDIS is set to ‘1’ the “Write Burst” command sequence is not allowed. Its use would cause unpredictable results.

ECCW
ECC Write Register **(1090_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
WCODE	[21:0]	rw	**Error Correction Write Code** 22-bit ECC code for the current 64-bit (for DFlash) or 256-bit (for PFlash) write buffer to be written into the assembly buffer instead of a generated ECC.
DECENCDIS	30	rw	**DF_EEPROM ECC Encoding Disable** 0_B The ECC code is automatically calculated. 1_B The ECC code is taken from WCODE.
PECENCDIS	31	rw	**PFlash ECC Encoding Disable** 0_B The ECC code is automatically calculated. 1_B The ECC code is taken from WCODE.
0	[29:22]	r	**Reserved** Always read as 0.

ECC Read Registers

For each Flash read path there is a separate Error Correction Code Read register. These allow disabling the ECC correction. The ECC decoding (i.e. the error detection) is not influenced. After reset ECC correction is enabled. If necessary the trap generation has to be separately disabled by MARP.TRAPDIS and MARD.TRAPDIS. Further on these registers allows to read the ECC code.

The ECC code of the last read access is stored in the RCODE field.

Note: After reading data with disabled ECC correction data buffers throughout the system (e.g. pre-fetch buffers in PMU and CPUs) may contain uncorrected data values. Therefore it is recommended to perform a reset to resume normal operation with ECC correction.

ECCRPp (p=0-0)
ECC Read Register PFp **(1094_H+p*4_H)** **Reset value: 0000 0000_H**
ECCRD
ECC Read Register DF **($10A4_H$)** **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECCORDIS	EDCERRINJ	0						RES		RCODE					
rw	rw	r						rh		rh					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCODE															
rh															

Field	Bits	Type	Description
RCODE	[21:0]	rh	**Error Correction Read Code** ECC code, read from the Flash read buffer with last data read operation.
RES	[23:22]	rh	**Reserved** Internal data.
EDCERRINJ	30	rw	**EDC Error Injection** Setting this bit enforces an error in the ECC error detection supervision circuit. This can be used to check the correct function of this circuit. This bit is only implemented for the PFlash read paths (i.e. ECCRPp). In ECCRD this bit is read-only 0. 0_B EDC logic operates normally. 1_B An error is injected into the EDC logic.
ECCORDIS	31	rw	**ECC Correction Disable** 0_B ECC correction for this read path is enabled. 1_B ECC correction for this read path is disabled.

Field	Bits	Type	Description
0	[29:24]	r	**Reserved** Always read as 0.

11.7.2.10 HSM Command Interface

The following registers constitute the HSM command interface together with its reserved address range for command sequences.

In devices without DF_HSM these registers are existing but not functional.

HSM Flash Status Register

HSMFSR
Flash Status Register **(1200_H)** **Reset Value: $0000\ 0004_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**SPND**	**EVER**	**PVER**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	rwh	rwh	rwh	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**SQER**	**OPER**	**DFPAGE**	**0**	**ERASE**	**PROG**	**0**	**0**	**0**	**0**	**D1BUSY**	**0**	**0**
r	r	r	rwh	rwh	rh	r	rwh	rwh	r	r	r	r	rh	r	r

Field	Bits	Type	Description
D1BUSY	2	rh	**Data Flash Bank 1 Busy**[1)] HW-controlled status flag. 0_B DFlash1 ready, not busy; DFlash1 in read mode. 1_B DFlash1 busy; DFlash1 not in read mode. Indication of busy state of DFlash bank 1 because of active execution of an operation initiated by HSM; DFlash1 busy state is also indicated in sleep mode; while in busy state the DFlash1 is not in read mode.

Field	Bits	Type	Description
PROG	7	rwh	**Programming State** [2)3)] HW-controlled status flag. 0_B There is no program operation requested or in progress or just finished. 1_B Programming operation (write page) requested or in action or finished. Set with last cycle of Write Page/Burst command sequence, cleared with Clear Status command (if not busy) or with power-on reset. If one BUSY flag is coincidently set, PROG indicates the type of busy state. If OPER is coincidently set, PROG indicates the type of erroneous operation. Otherwise, PROG indicates, that operation is still requested or finished. Can be also cleared by writing ‘1’ to it.
ERASE	8	rwh	**Erase State** [2)3)] HW-controlled status flag. 0_B There is no erase operation requested or in progress or just finished 1_B Erase operation requested or in action or finished. Set with last cycle of Erase/Verify command sequence, cleared with Clear Status command (if not busy) or with power-on reset. Indications are analogous to PROG flag. Can be also cleared by writing ‘1’ to it.
DFPAGE	10	rh	**Data Flash in Page Mode**[1)4)] HW-controlled status flag. 0_B Data Flash not in page mode 1_B Data Flash in page mode Set with Enter Page Mode for DFlash, cleared with Write Page command. *Note: Concurrent page and read modes are allowed*

Field	Bits	Type	Description
OPER	11	rwh	**Flash Operation Error[2)3)4)]** 0_B No operation error. 1_B Flash array operation aborted, because of a Flash array failure, e.g. an ECC error in microcode. This bit is not cleared with application reset, but with power-on reset. Registered status bit; must be cleared per command or by writing ‘1’.
SQER	12	rwh	**Command Sequence Error[1)2)4)]** 0_B No sequence error 1_B Command state machine operation unsuccessful because of improper address or command sequence. A sequence error is not indicated if the Reset to Read command aborts a command sequence. Registered status bit; must be cleared per command or by writing ‘1’.
PVER	25	rwh	**Program Verify Error[1)2)4)]** A verify error was reported during a Flash program operation. 0_B The page is correctly programmed. All bits have full expected quality. 1_B A program verify error has been detected. Full quality of all bits cannot be guaranteed. Registered status bit; must be cleared per command or writing ‘1’.
EVER	26	rwh	**Erase Verify Error[1)2)4)]** A verify error was reported during a Flash erase operation. 0_B The sector is correctly erased. All erased bits have full expected quality. 1_B An erase verify error has been detected. Full quality erased bits cannot be guaranteed. Registered status bit; must be cleared per command or writing ‘1’.

Field	Bits	Type	Description
SPND	27	rwh	**Operation Suspended**[3)] Requested by HSMMARD.SPND a program or erase operation is suspended. 0_B No Flash operation is suspended. 1_B Suspended Flash operation. Can be also cleared by writing '1' to it to clean up after a reset that killed a suspended operation context.
0	[31:28], [24:13], 9, [6:3], 1, 0	r	**Reserved** Read zero, no write

1) Cleared with application reset
2) Cleared with command "Clear Status"
3) Cleared with power-on reset (PORST)
4) Cleared with command "Reset to Read"

Note: The xBUSY flags cannot be cleared with the "Clear Status" command or with the "Reset to Read" command. These flags are controlled by HW.

HSM Flash Configuration Register

HSMFCON
HSM Flash Configuration Register (1204_H) **Reset value: $0000\ 0001_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EOBM	**0**					**SQERM**	**VOPERM**	**0**							
rw	r					rw	rw	r							

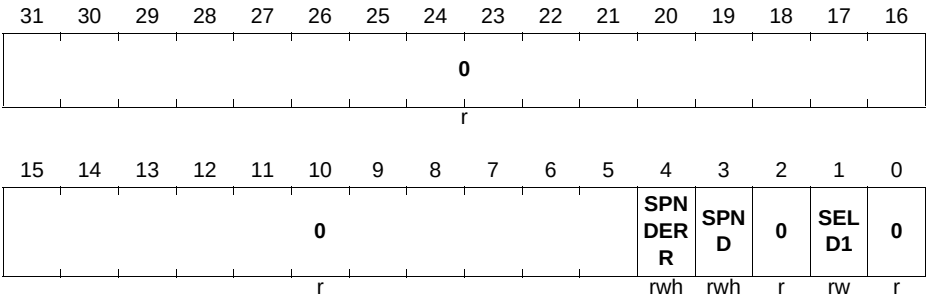
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														**LCK**	**HSMUCB**
r														rwh	

Field	Bits	Type	Description
LCKHSMUCB	[1:0]	rwh	**Lock Access to UCB_HSMCFG** Trap door register. This field can only be written to the "Locked" state. Other writes are ignored (see also **Page 11-31**) 01_B **Unlocked**, Reads by HSM to UCB_HSMCFG allowed. $..._B$ **Locked**, Reads to UCB_HSMCFG forbidden.
VOPERM	24	rw	**Verify and Operation Error Interrupt Mask** 0_B Interrupt not enabled 1_B HSM Flash interrupt because of Verify Error or Operation Error in Flash array (FSI) is enabled
SQERM	25	rw	**Command Sequence Error Interrupt Mask** 0_B Interrupt not enabled 1_B HSM Flash interrupt because of Sequence Error is enabled
EOBM	31	rw	**End of Busy Interrupt Mask** 0_B Interrupt disabled. 1_B EOB interrupt is enabled.
0	[30:26], [23:2]	r	**Reserved** Write 0; Read 0.

Margin Check Control HSM DFlash and Suspend

HSMMARD
Margin Control Register HSM DFlash

(1208_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
SELD1	1	rw	**HSM DFLASH Bank Selection** Bank DF1 (HSM DFlash) is selected for reading with HMARGIN. 0_B Bank DF1 is read with the standard (default) margin independent of MARD.HMARGIN. 1_B Bank DF1 is read with the margin selected by MARD.HMARGIN.
SPND	3	rwh	**Suspend** 0_B No suspend requested or pending. 1_B Suspension of the ongoing program/erase process is requested or pending.
SPNDERR	4	rwh	**Suspend Error** 0_B No suspend error. 1_B Last suspend request via HSMMARD.SPND failed (see **Chapter 11.5.4.5**). Can be cleared by writing ‘1’.
0	0, 2, [15:4], [31:16]	r	**Reserved** Always read as 0; should be written with 0.

11.7.2.11 HSM Requested Read Interface

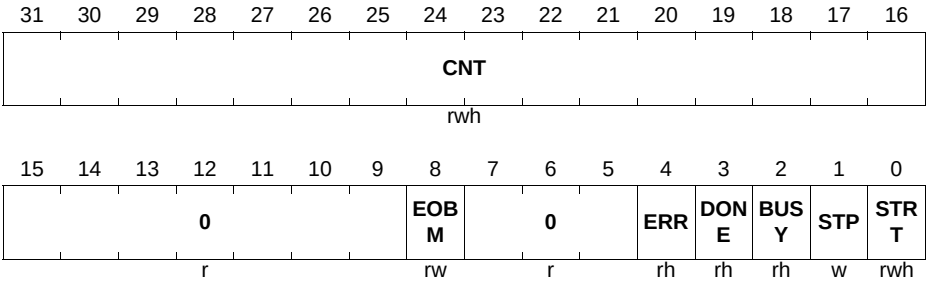
The following registers can be used by the HSM to control its requested read interface and perform data transfers (see **Chapter 11.5.3.4**) from the HSMx sectors.

In devices without DF_HSM these registers are existing but not functional.

HSM Requested Read Control

HSMRRCT
Requested Read Control Register HSM

(120C$_H$) Reset Value: 0000 0000$_H$



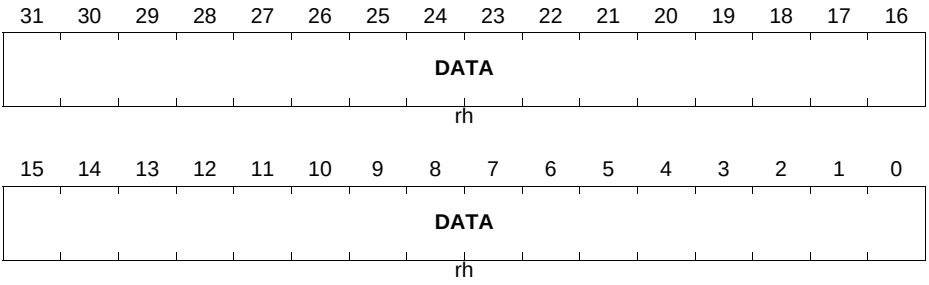
Field	Bits	Type	Description
STRT	0	rwh	**Start Request** Writing ‘1’ to this bit starts the read process as configured by HSMRRAD and HSMRRCT.CNT. This bit is cleared by the PMU as soon the reading process starts executing.
STP	1	w	**Stop** Stops the read process.
BUSY	2	rh	**Flash Read Busy** 0_B The HSMRRD registers contain the data addressed by HSMRRAD (data ready). 1_B The Flash is busy reading the data addressed by HSMRRAD.
DONE	3	rh	**Flash Read Done** 0_B The Flash read has not finished. 1_B The Flash read has finished, data is available.
ERR	4	rh	**Error** Cleared when starting a sequence with STRT, set when an error is detected. 0_B No error occurred in this sequence. 1_B Error occurred.

Field	Bits	Type	Description
EOBM	8	rw	**End of Busy Interrupt Mask** Interrupt for HSMRRCT.BUSY. 0_B Interrupt disabled. 1_B EOB interrupt is enabled.
CNT	[31:16]	rwh	**Count** Defines number of remaining data transfers.
0	[15:9], [7:5]	r	**Reserved** Always read as 0; should be written with 0.

HSM Requested Read Data Registers

HSMRRDx (x=0-1)
HSM Requested Read Data Register x

(1210_H+x*4_H) **Reset Value: 0000 0000_H**

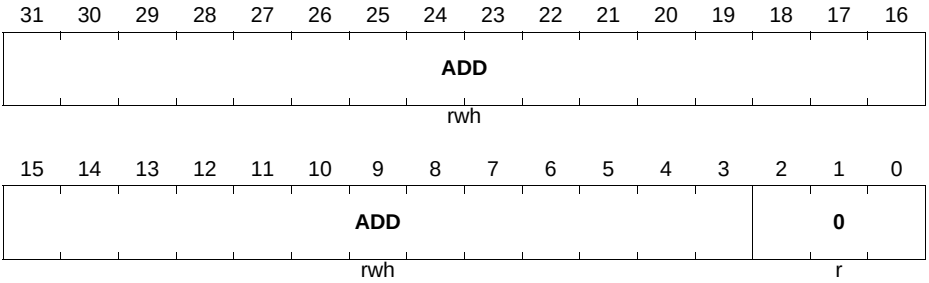


Field	Bits	Type	Description
DATA	[31:0]	rh	**Read Data** Read data.

HSM Requested Read Address Register

HSMRRAD
HSM Requested Read Address Register

(1218$_H$) **Reset Value: 0000 0000$_H$**



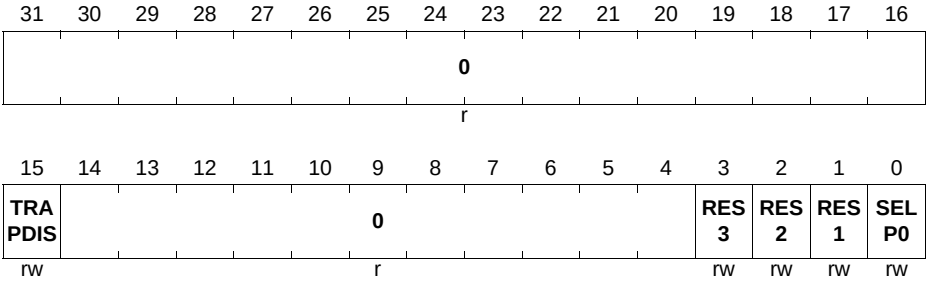
Field	Bits	Type	Description
ADD	[31:3]	rwh	**Address** Start address/current address. Must point to the DF1 address range starting at AN_DFlash_B1. The address range at AN_DFlash_B1F is not allowed (sets HSMRRCT.ERR). Can be written by software to define the next read address. Is automatically incremented by the requested read hardware when finishing the last read access. In case both writes happen concurrently the hardware update is performed, the software write is ignored.
0	[2:0]	r	**Reserved** Always read as 0; should be written with 0.

11.7.2.12 Margin Check Control

Note: Although uncorrectable error traps are disabled with reset, the traps are enabled by the startup SW (firmware) in Boot ROM before Boot ROM exit.

Margin Check Control PFlash

MARP
Margin Control Register PFlash (10A8$_H$) Reset Value: 0000 8000$_H$



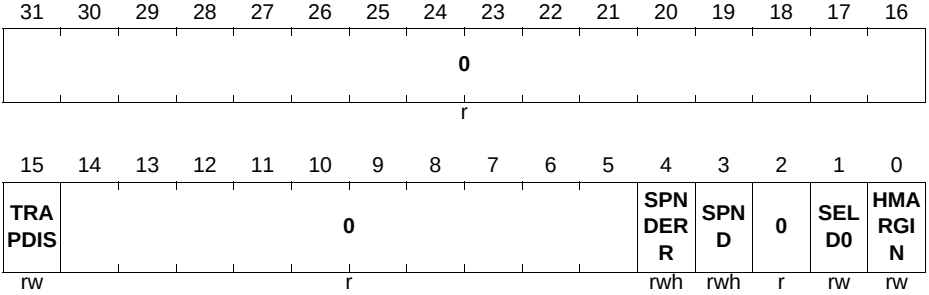
Field	Bits	Type	Description
SELPi(i=0-0)	i	rw	**PFLASH Bank PFi Selection** Bank PFi is selected for reading with MARD.HMARGIN. 0_B Bank PFi is read with the standard (default) margin independent of HMARGIN. 1_B Bank PFi is read with the margin selected by HMARGIN.
RES1	1	rw	**Reserved** Reserved for PFLASH Bank Selection SELP1.
RES2	2	rw	**Reserved** Reserved for PFLASH Bank Selection SELP2.
RES3	3	rw	**Reserved** Reserved for PFLASH Bank Selection SELP3.
TRAPDIS	15	rw	**PFLASH Uncorrectable Bit Error Trap Disable** 0_B If an uncorrectable error occurs in PFLASH (setting FSR.PFMBER), a bus error trap is generated[1)]. 1_B The uncorrectable error trap is disabled.

Field	Bits	Type	Description
0	[14:4], [31:16]	r	**Reserved** Always read as 0; should be written with 0.

1) After Boot ROM exit, uncorrectable bit error traps are enabled (TRAPDIS = 0_B).

Margin Check Control DFlash and Suspend

MARD
Margin Control Register DFlash (10AC$_H$) **Reset Value: 0000 8000$_H$**



Field	Bits	Type	Description
HMARGIN	0	rw	**Hard Margin Selection** Banks selected by MARD.SELi = 1 or MARP.SELi = 1 are read with: 0_B **Tight0**, Tight margin for 0 (low) level. Suboptimal 0-bits are read as 1s. 1_B **Tight1**, Tight margin for 1 (high) level. Suboptimal 1-bits are read as 0s. The concrete margin values are restored from the configuration sector and are determined by Infineon.
SELD0	1	rw	**DFLASH Bank Selection** Bank DF0 is selected for reading with HMARGIN. 0_B Bank DF0 is read with the standard (default) margin independent of HMARGIN. 1_B Bank DF0 is read with the margin selected by HMARGIN.

Field	Bits	Type	Description
SPND	3	rwh	**Suspend** 0_B No suspend requested or pending. 1_B Suspension of the ongoing program/erase process is requested or pending.
SPNDERR	4	rwh	**Suspend Error** 0_B No suspend error. 1_B Last suspend request via MARD.SPND failed (see **Chapter 11.5.4.5**). Can be cleared by writing ‘1’.
TRAPDIS	15	rw	**DFLASH Uncorrectable Bit Error Trap Disable** 0_B If an uncorrectable error occurs in DFLASH (setting FSR.DFMBER), a bus error trap is generated[1)]. 1_B The uncorrectable error trap is disabled.
0	2, [14:5], [31:16]	r	**Reserved** Always read as 0; should be written with 0.

1) After Boot ROM exit, uncorrectable bit error traps are enabled (TRAPDIS = 0_B).

11.7.2.13 Corrected Bits Address Buffer (CBAB)

When data is read from PFlash and the ECC decoder detects correctable bit errors those addresses are stored in the “corrected bits address buffer”. Each address is only entered once. Depending on its configuration 1-bit and 2-bit errors or only 2-bit errors are entered.

When CBAB becomes full the SMU is informed which can trigger an interrupt.

When the CBAB is full no further addresses are added. With CBABCFGp.CLR or CBABTOPp.CLR entries can be emptied.

There exists one CBAB per PFlash “p” SRI port.

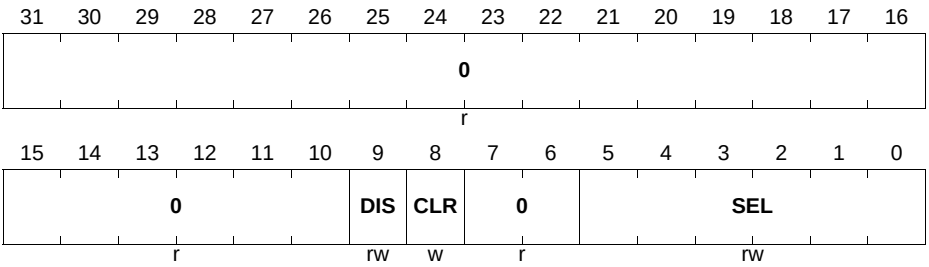
Each CBAB instance implements ten entries.

The CBAB is called in safety documentation “PFLASH Monitor”.

CBAB Configuration

Reset: power-on reset.

CBABCFGp (p=0-0)
CBAB Configuration Port p **($10B4_H+p*C_H$)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
SEL	[5:0]	rw	**Select Bit-Errors** This bit controls which type of errors are entered into the CBAB. 000000_BNo errors enter the CBAB. 000001_BCorrected single-bit errors enter the CBAB. 000010_BCorrected double-bit errors enter the CBAB. 000011_BCorrected single-bit and double-bit errors enter the CBAB. others: Reserved values.

Field	Bits	Type	Description
CLR	8	w	**Clear** Clears the complete CBABp.
DIS	9	rw	**Disable** This bit allows to disable this CBAB. This reduces the power consumption. The content of the CBAB stays unchanged. *Note: Due to race conditions switching this bit concurrently to Flash read accesses can cause an erroneous address to become registered twice.*
0	[7:6], [31:10]	r	**Reserved** Always read as 0; should be written with 0.

CBAB Status

Reset: power-on reset.

CBABSTATp (p=0-0)
CBAB Status Port p **($10B8_H+p*C_H$)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						**VLD 9**	**VLD 8**	**VLD 7**	**VLD 6**	**VLD 5**	**VLD 4**	**VLD 3**	**VLD 2**	**VLD 1**	**VLD 0**
r						rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

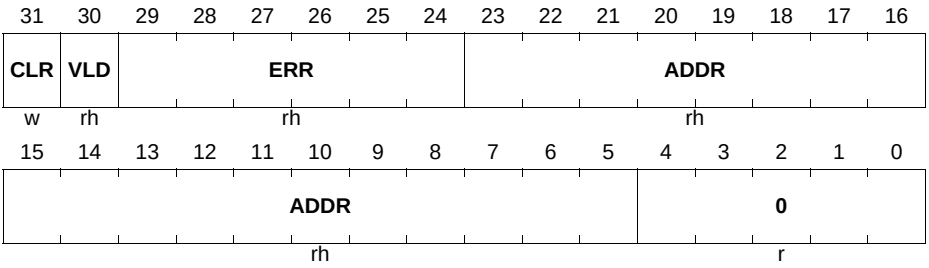
Field	Bits	Type	Description
VLDx (x=0-9)	x	rh	**Filling Level** Each VLDx indicates a valid entry. The complete set of VLD flags operates as thermometer code. Entry 0 is the top entry which is read in CBABTOP. The entry with the highest index number was entered last.
0	[31:10]	r	**Reserved** Always read as 0; should be written with 0.

CBAB FIFO Top Entry

This register displays the information of the top-most entry of the CBAB FIFO.

Reset: power-on reset (for the complete FIFO content).

CBABTOPp (p=0-0)
CBAB FIFO TOP Entry Port p ($10BC_H+p*C_H$) Reset Value: 0000 0000_H



Field	Bits	Type	Description
CLR	31	w	**Clear** Write-only trigger bit. Reading returns 0. 1_B Clears valid flag “VLD” of this entry and removes it from the FIFO. 0_B No operation.
VLD	30	rh	**Valid** Entry is valid.
ERR	[29:24]	rh	**Error Type** This field contains the error type found at ADDR. The bits correspond to CBABCFG.SEL. 000000_BNo error (reserved). 000001_BSingle-bit error. 000010_BDouble-bit error. others: Reserved.
ADDR	[23:5]	rh	**Address** Captured address (bits 23:5 of the SRI bus address) with detected error.
0	[4:0]	r	**Reserved** Always read as 0; should be written with 0.

11.7.2.14 Uncorrectable Bits Address Buffer (UBAB)

When data is read from PFlash and the ECC decoder detects uncorrectable bit errors those addresses are stored in the “uncorrected bits address buffer”. Each address is only entered once. Depending on its configuration selected failures can be entered.

When the UBAB is full no further addresses are added. With UBABCFGp.CLR or UBABTOPp.CLR entries can be emptied.

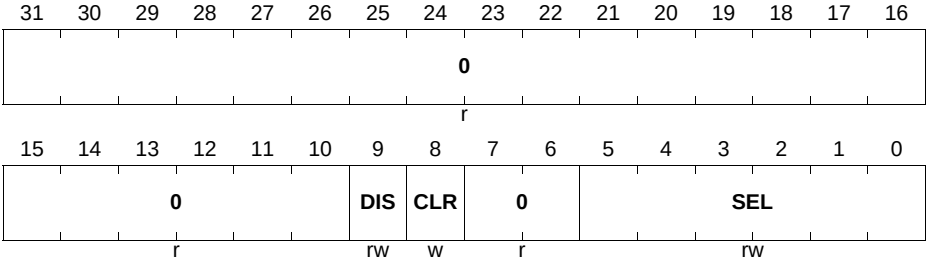
There exists one UBAB per PFlash “p” SRI port.

Each UBAB instance implements one entry.

UBAB Configuration

Reset: power-on reset.

UBABCFGp (p=0-0)
UBAB Configuration Port p **($10E4_H$+p*C_H)** **Reset Value: 0000 0000_H**



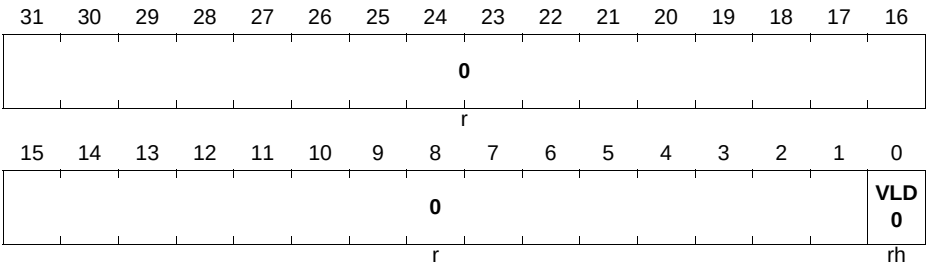
Field	Bits	Type	Description
SEL	[5:0]	rw	**Select Bit-Errors** This bit controls which type of errors are entered into the UBAB. 000000_BNo errors enter the UBAB. 000100_BDetected uncorrectable bit-errors (3-bit or more, address errors, all-0, all-1 enter the UBAB. others: Reserved values.
CLR	8	w	**Clear** Clears the complete UBABp.

Field	Bits	Type	Description
DIS	9	rw	**Disable** This bit allows to disable this UBAB. This reduces the power consumption. The content of the UBAB stays unchanged.
0	[31:10], [7:6]	r	**Reserved** Always read as 0; should be written with 0.

UBAB Status

Reset: power-on reset.

UBABSTATp (p=0-0)
UBAB Status Port p **($10E8_H+p*C_H$)** **Reset Value: 0000 0000$_H$**



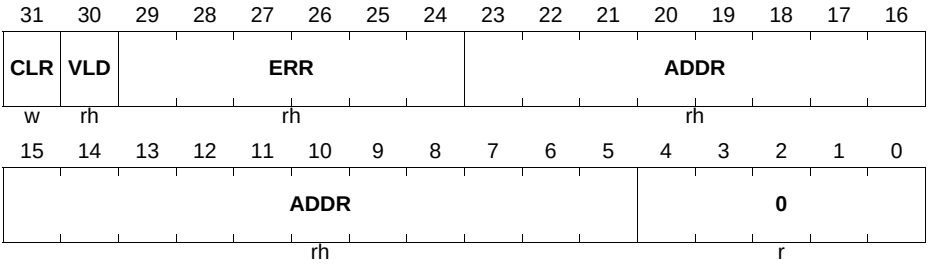
Field	Bits	Type	Description
VLD0	0	rh	**Filling Level** Each VLDi indicates a valid entry. The complete set of VLD flags operates as thermometer code. Entry 0 is the top entry which is read in UBABTOP. The entry with the highest index number was entered last.
0	[31:1]	r	**Reserved** Always read as 0; should be written with 0.

UBAB FIFO Top Entry

This register displays the information of the top-most entry of the UBAB FIFO.

Reset: power-on reset (for the complete FIFO content).

UBABTOPp (p=0-0)
UBAB FIFO TOP Entry Port p ($10EC_H+p*C_H$) Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
CLR	31	w	**Clear** Write-only trigger bit. Reading returns 0. 1_B Clears valid flag “VLD” of this entry and removes it from the FIFO. 0_B No operation.
VLD	30	rh	**Valid** Entry is valid.
ERR	[29:24]	rh	**Error Type** This field contains the error type found at ADDR. The bits correspond to UBABCFG.SEL. 000000_BNo error (reserved). $XXX100_B$Uncorrectable error (3-bit or more, addressing error, all-0, all-1). others: Reserved.
ADDR	[23:5]	rh	**Address** Captured address (bits 23:5 of the SRI bus address) with detected error.
0	[4:0]	r	**Reserved** Always read as 0; should be written with 0.

11.7.2.15 Direct Flash Communication

These registers enable direct communication of software (e.g. a Flash driver running on TriCore) with the FSI specifically with its µCode.

Because of design particularities byte wide write accesses are not supported and cause unpredictable results. Only half-word (16-bit) wide writes shall be performed.

Read and write accesses during ongoing Flash operations (FSR shows BUSY flags) is forbidden. The PMU returns a bus error.

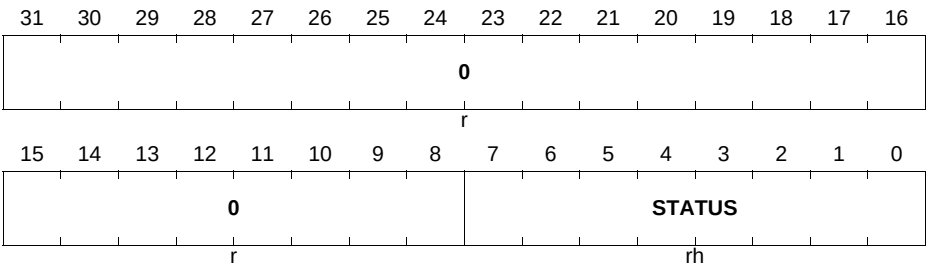
These registers shall be read or written in exceptional cases only when demanded by IFX for extended functions.

The behavior of these registers in specific Flash modes is special:

- During all FSI operations including Flash startup: the STATUS fields can deliver unpredictable data.
- When the Flash is in sleep mode: write and read accesses fail with a bus error.

Flash Communication Register 0

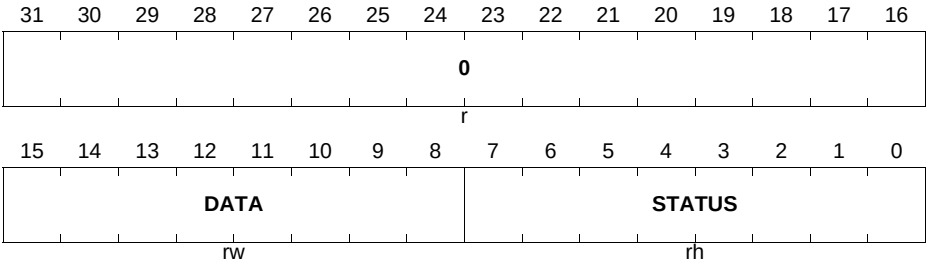
COMM0
FSI Communication Register 0 (0000_H) Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
STATUS	[7:0]	rh	**Status** This field can be only written by the FSI.
0	[31:8]	r	**Reserved** Always read as 0.

Flash Communication Register

COMMx (x=1-2)
FSI Communication Register x **(0000_H+4*x)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
STATUS	[7:0]	rh	**Status** This field can be only written by the FSI.
DATA	[15:8]	rwh	**Data** This field can be only written by software not by the FSI.
0	[31:16]	r	**Reserved** Always read as 0.

11.8 Application Hints

The following application hints should support the user in using the PMU and Flash optimally.

11.8.1 Changes with Respect to Audo Families Audo-NG/F/S/Max

The following changes of the PMU with respect to Audo-NG/F/S/Max products are highlighted. With “Audo-Max” are changes noted with respect to SRI based Audo-Max products. The tag “All” indicates changes with respect to all Audo product generations:

- All: Flash cell concept is changed enabling many significant improvements:
  - Performance of programming significantly enhanced.
  - Burst programming feature with even higher programming throughput.
  - No over-erased state anymore.
  - Sector structure of PFlash enhanced with more small Flash sectors.
  - EEPROM emulation in DFlash improved by smaller pages.
  - Enhanced retention and endurance figures.
  - All logical sectors of PFlash (see **Table 11-2**) have the same endurance. “Second class” logical sectors with only 100 cycle endurance are not existing anymore.
- All: Changed PMU and FSI architecture (enabled by the increased programming performance). Now one FSI and PMU can serve several PFlash banks in addition to the DFlash banks.
- All: Due to this one PMU and FSI has to support several independent read paths with independent bus interfaces to the assigned PFlash banks.
- All: Changed command sequences. In Audo the SA and PA arguments were transmitted in the address phase, thus these command cycles had to be written to the target Flash range. Now SA and PA are transmitted in the data phase. Only writes into the DFlash address range are interpreted as command sequences.
- Audo-NG/F/S: Enhanced Flash read buffer concept with master specific read buffers.
- Audo-NG/F/S: Changed bus interface from LMB to SRI.
- All: Changed supply concept supporting EVR. Dedicated programming modes for 3.3V only supply or 5V supply.
- Audo-NG/F/S: The wait-state settings in FCON don’t relate by default to the bus clock as in Audo-NG/F/S (i.e. the LMB clock) but to the FSI and FSI2 clocks. These clocks have their own divider from the PLL clock which need to be configured in SCU_CCUCON0.FSIDIV and SCU_CCUCON0.FSI2DIV. They have to be configured to have a relation $f_{SRI}/f_{FSI(2)}$ of n/1.
- All: Explicit suspend to read feature replacing automatic “program while erase” of the Audo generation.
- Audo-NG/F/S: The OVRAM and the EMEM were moved to a different module “LMU”.
- All: The register field FCON.STALL configures the behavior when reading busy Flash banks. Either “ready” is suppressed or a bus error is generated.
- Audo-NG/F/S: TP: changed addresses RTPWT and RPSSW.

- Audo-NG/F/S: Programming and reading data with disabled ECC generation and correction allows customers to program data with invalid ECC.
- Audo-NG/F/S: Error flags in FSR can be selectively cleared to enable easier flag handling for concurrent operations.
- Audo-Max: Dropped extended status register XSFR.
- All: Protection fields assembled in new register FPRO.
- All: Changed protection concept:
  - Only one protection “user” in PFlash.
  - Separate protection with password for DFlash.
  - Password width increased from 64 bits to 256 bits.
  - More flexible OTP configuration that allows adding OTP.
  - New write-once protection enabling “automatic” OTP protection.
  - All UCBs realized as part of DFlash instead of PFlash.
  - Confirmation concept for UCB blocks changed. Now an erased confirmation code activates complete protection and locks the UCB.
  - New password protected debug lock.
- All: Support for HSM:
  - Dedicated protection for HSM code sectors.
  - HSM related UCBs for configuring HSM Flash protection.
- All: New erase counters.
- All: Changed base addresses of all Flash modules.
- All: Changed base address of BootROM.
- All: TEC-QED ECC algorithm in DFlash.
- All: DEC-TED ECC algorithm in PFlash with specific safety features for 99% white noise coverage, detection of all-0/all-1 inducing faults and addressing errors.
  - As in Audo-Max the ECC algorithm can be switched between safety and non-safety mode.
  - Attention: switching the algorithm with FCON.NSAFECC may only be done while no Flash accesses are performed. This feature may conflict with Audo-F Flash driver software.
- All: New “Erase Verify” functionality.
- All: New multiple sector erase command sequence “Erase Logical Sector Range”.
- All: Extended safety protection of registers with ACCEN and “Safety ENDINIT”.
- All: FIFOs that store uniquely addresses in PFlash read with corrected and uncorrectable bit errors.

11.8.2 Performing Flash Operations

This section offers advice for programming command sequences.

General Advice

- Please remember to disable the Safety ENDINIT protection before executing program and erase commands for the PFlash (see **Chapter 11.5.5.2**).

- Code that performs PFlash programming or erasing should not be executed from the same PFlash.
- The command cycles shall address the non-cached address range of the Flash (otherwise the data may stay in the cache or could be received by the PMU in an incorrect order).
- The non-cached Flash range is not regarded as peripheral address space by the TriCore. This means that a read (notably reading the FSR for checking the flags) placed behind the last command cycle write in program order might be executed before this last write.
  The FSR flags xFPAGE, PROG, ERASE can be used to ensure that polling for a cleared BUSY starts only after the command was accepted by the PMU.
  Additionally it is recommended to place a “DSYNC” instruction between a command sequence and read accesses to depending data including affected registers.
- The caches (data cache in DMI “DCache” and program cache in PMI “PCACHE”) as well as the data line buffer in the DMI “DLB” are not automatically invalidated nor updated after changing Flash content by erasing or programming. It is therefore recommended to either invalidate them actively or read the Flash content via the non-cached address range. Otherwise old data might be delivered from these buffers.
  The DMI cache and line buffer can be invalidated by writing OVCCON.DCINVAL to ‘1’. The PCACHE can be invalidated by writing PCON1.PCINV to ‘1’. All buffers are invalidated by a reset.
- The PMU and Flash work with the SRI, FSI and FSI2 clocks. When changing the divider values of these clocks in SCU_CCUCON0 the following rules apply:
  – The only allowed PMU/Flash “operation” when switching is reading from Flash memory. Programming or erasing the Flash is forbidden.
  – It must be ensured that before, during and after the divider change the configured number of Flash wait-cycles is sufficient for the selected clock frequency. Remember that the wait-cycles are counted with f_{FSI2} for PFlash and f_{FSI} for DFlash.
  – After System Resets and Power-On Resets the wait cycle values are configured to a value only sufficient for the clock frequencies used during startup (i.e. 100 MHz). So basically always changes to the clock configuration must be preceded by changes to the wait cycles.

Sequence for Programming

The following sequence is the most defensive one for programming a page. It is however acceptable to skip some checks when the programmed data is later verified:

- “Clear Status” to clear flags.
- “Enter Page Mode”.
- DSYNC.
- Wait until FSR.xFPAGE = ‘1’ or fail if FSR.SQER = ‘1’ or FSR.PROER = ‘1’.
- Repeat “Load Page” until the page is filled.

- “Write Page”.
- DSYNC.
- Wait until FSR.PROG = ‘1’ or fail if FSR.SQER = ‘1’ or FSR.PROER = ‘1’.
- Wait until FSR.xBUSY = ‘0’ or enable the interrupt.
  – While FSR.xBUSY is ‘1’ the flags FSR.OPER can be checked for ‘1’ as abort criterion to protect against hardware failures causing BUSY to stay ‘1’.
- Check for FSR.PVER flag.
- Fail if FSR.OPER = ‘1’.
- Recommended: check programmed content, evaluate FSR.xMBER and possibly count correctable errors.
- Clear error flags either with “Clear Status” or by directly writing to FSR.

Sequence for Erasing

The following sequence is the most defensive one for erasing a range of sectors. It is however acceptable to skip some checks when the programmed data is verified after programming (or the flag FSR.PVER is checked as described above):

- “Clear Status” to clear flags.
- “Erase Logical Sector Range”.
- DSYNC.
- Wait until FSR.ERASE = ‘1’ or fail if FSR.SQER = ‘1’ or FSR.PROER = ‘1’.
- Wait until FSR.xBUSY = ‘0’ or enable the interrupt.
  – While FSR.xBUSY is ‘1’ the flags FSR.OPER can be checked for ‘1’ as abort criterion to protect against hardware failures causing BUSY to stay ‘1’.
- Check for FSR.EVER/PVER flags.
- Fail if FSR.OPER = ‘1’.
- Clear error flags either with “Clear Status” or by directly writing to FSR.

An analog sequence can be used for “Verify Erased Logical Sector Range”.

11.8.3 EEPROM Emulation With DFlash

The term “EEPROM emulation” designates an algorithm with the following features:

- It increases the effective endurance by spreading the EEPROM write accesses over a larger range of Flash memory.
- It ensures that all Flash cells incur a similar number of cycles independent of the update frequency of the EEPROM data (“wear levelling”).
- It manages the allocation of EEPROM data to Flash ranges so that stale data can be erased.

A popular emulation algorithm is the “two sector” emulation, which splits the complete DFlash in two equally sized ranges:

- The EEPROM writes are performed in one range, the “active” one. The other range stays erased.

- The data is stored so that several writes of the same EEPROM address can be performed in one DFlash range. The position of the latest data version and the original EEPROM address can be determined from the address in Flash and additional administrative data (e.g. all data is pushed to a stack and the pushed data contains the EEPROM address).
- At a certain range fill level a range switch is performed. This moves all active data to the erased range. After that the active range is erased and the other range becomes the active one.

With the 6 erasable DFlash ranges of the TC21x/TC22x/TC23x other more efficient algorithms are possible. However the algorithm must ensure that all sectors are used equally, e.g. it is not allowed to keep static data in certain sectors and often updated data in other sectors. The sectors with static data would incur too many disturbs.

One specific requirement for an EEPROM emulation algorithm is the resistance against aborts during its operation. In any case the algorithm must be able after device startup to recover without data loss, determine the active data and resume operation.

In many applications even resistance against aborted Flash processes (program, erase) is needed. See details in **Chapter 11.8.5.2**.

11.8.3.1 Robust EEPROM Emulation

A key requirement for an EEPROM emulation algorithm is the reliability of the stored data. The DFlash with its TEC-QED ECC algorithm protects perfectly against bit or bit-line oriented failures.

However, ECC mechanisms do not protect against word-line oriented failures. Word-line failure modes may result in a complete word-line which can no longer be programmed or erased, and the data within the failed word-line may not be able to be read correctly. Word-line failures are most likely to occur during the high voltage conditions present during programming or erase operations. Therefore, a robust EEPROM emulation algorithm must tolerate word-line failures, and must protect against data loss in case the word-line fails during an erase or programming operation.

The following steps shall be followed to achieve the necessary robustness:

- Before programming a page save the content of all other pages on the same word-line that contain active data to SRAM.
- Program the new page and compare the content of this page and of the saved pages with their reference data. This can be done with normal read margins. Ignore correctable bit-errors.
- If the data comparison fails or the programming returned with PVER error program this page and the saved content of the other pages to a different word-line.
- This procedure can be repeated if the data comparison fails again. The number of repetitions should be limited (e.g. to 3) in case the programming fails because of out-of-spec operating conditions.

- Word-line oriented fails can also have the effect that the affected word-lines can not be erased anymore, yet the contents of the word-line data could still be read without any error indication. Sequence counters for the data blocks or other means must be used to identify old data blocks in word-lines with this type of failure mode.

For the TC21x/TC22x/TC23x this robust EEPROM algorithm is required for the usage of the DFlash.

Due to the specificity of each application the appropriate usage and implementation of these measures must be chosen according to the context of the application.

11.8.4 Handling Errors

The earlier sections described shortly the functionality of “error indicating” bits in the flash status register **FSR**. This section elaborates on this with more in-depth explanation of the error conditions and recommendations how these should be handled by customer software.

11.8.4.1 Handling Errors During Operation

This first part handles error conditions occurring during operation (i.e. after issuing command sequences) and the second part (**Section 11.8.4.2**) error conditions detected during startup.

SQER “Sequence Error”

Fault conditions:

- Improper command cycle address or data, i.e. incorrect command sequence.
- New “Enter Page” in Page Mode.
- “Load Page” and not in Page Mode.
- “Load Page” results in buffer overflow.
- “Load Page” with mixed 32/64-bit transfers.
- First “Load Page” addresses 2. word.
- “Write Page” with buffer underflow.
- “Write Page” and not in Page Mode.
- “Write Page” to unavailable Flash range.
- Command sequence with address not pointing to a legal start address (e.g. page, UCB or sector).
- “Write Page Once” targeting DFlash.
- Byte transfer to password or data.
- “Erase Logical Sector Range” or “Verify Erased Logical Sector Range” with range leaving a physical sector.
- “Resume Prog/Erase” with arguments not matching the suspended command.
- “Resume Prog/Erase” when there is no suspended operation.

- Any programming or erase command when there is a suspended programming command.
- Any erase command when there is a suspended erase command, including “Verify Erased Logical Sector Range”.
- Programming to the target range of a suspended erase command.
- Unsupported command sequence for the HSM command interpreter.

New state:

Read mode is entered with following exceptions:

- “Enter Page” in Page Mode re-enters Page Mode.
- “Write Page” with buffer underflow is executed.
- After “Load Page” causing a buffer overflow the Page Mode is not left, a following “Write Page” is executed.

Proposed handling by software:

Usually this bit is only set due to a bug in the software. Therefore in development code the responsible error tracer should be notified. In production code this error will not occur. It is however possible to clear this flag with “Clear Status” or “Reset to Read” and simply issue the corrected command sequence again.

OPER “Operation Error”

Fault conditions:

ECC double-bit error detected in Flash microcode SRAM during an operation or the microcode has detected another significant failure. This can be a transient event due to alpha-particles or illegal operating conditions or it is a permanent error due to a hardware defect. This situation will practically not occur.

Attention: these bits can also be set during startup (see **Chapter 11.8.4.2**).

New state:

The Flash operation is aborted, the BUSY flag is cleared and read mode is entered.

Proposed handling by software:

The last operation can be determined from the PROG and ERASE flags. In case of an erase operation the affected physical sector must be assumed to be in an invalid state, in case of a program operation only the affected page. Other sectors can still be read. New Flash commands (Write*, Erase*, Verify*, Resume*) must not be issued before the next system or power-on reset. “Clear Status” can be used to clear the OPER flag, however this doesn’t resolve the corrupted state of the FSI.

Attention: New FSI operations in this state can cause corruption of Flash content, which might even prevent the device from booting.

The next system or power-on reset performs a new Flash startup with initialization of the microcode SRAM. A power-on reset clears additionally the OPER flag. The application must determine from the context which operation failed and react accordingly. Mostly

erasing the addressed sector and re-programming its data is most appropriate. If a “Program Page” command was affected and the sector can not be erased (e.g. in Flash EEPROM emulation) the word-line could be invalidated if needed by marking it with all-one data and the data could be programmed to another empty word-line.

Only in case of a defective microcode SRAM the next program or erase operation will incur again this error.

PROER “Protection Error”

Fault conditions:

- Password failure.
- Erase/Write to protected sector.
- Erase/Write of protected UCB.
- “Verify Erased Logical Sector Range” to blocked addresses.
- Program or erase targeting PFlash with active Safety ENDINIT protection.

New state:

Read mode is entered. The protection violating command is not executed.

Proposed handling by software:

Usually this bit is only set during runtime due to a bug in the software. In case of a password failure a reset must be performed in the other cases the flag can be cleared with “Clear Status” or “Reset to Read”. After that the corrected sequence can be executed.

EVER “Erase Verify Error”

Fault conditions:

This flag is set by the erase commands when they don’t achieve an optimum result.

New state:

No state change. Just the bit is set.

Proposed handling by software:

This bit should be cleared with “Clear Status” or “Reset to Read”.

The operating conditions should be checked. The following advice assumes correct operating conditions and a correctly configured device.

In the PFlash an EVER can be regarded as device failure.

In the DFlash an EVER is a warning which doesn’t require immediate action. The robust EEPROM emulation will detect during programming the corresponding word-lines and “jump” over them.

*Note: Even when this flag is ignored it is recommended to clear it. Otherwise all following operations — including “sleep” — could trigger an interrupt even when they are successful (see **Chapter 11.5.7**, interrupt because of verify error).*

PVER "Program Verify Error"

Fault conditions:

This flag is set by the program commands when they don't achieve an optimum result.

New state:

No state change. Just the bit is set.

Proposed handling by software:

This bit should be cleared with "Clear Status" or "Reset to Read".

The operating conditions should be checked. The following advice assumes correct operating conditions and a correctly configured device.

In the PFlash a PVER can be regarded as device failure. However due to the high correction capability of the ECC customers may choose to check the amount of corrected and uncorrectable errors and decide on this basis if the device can stay in operation.

In the DFlash a PVER is a signal for the robust EEPROM emulation that programming on the current word-line failed. The algorithm has to "jump" over this word-line and programs its data to the next word-line.

*Note: Even when this flag is ignored it is recommended to clear it. Otherwise all following operations — including "sleep" — could trigger an interrupt even when they are successful (see **Chapter 11.5.7**, interrupt because of verify error).*

PFSBER, PFDBER, DFSBER, DFDBER, DFTBER "Single/Double/Triple-Bit Error"

Fault conditions:

When reading data or fetching code from PFlash or DFlash the ECC evaluation detected an error which was corrected.

This flag is a warning indication and not an error. A certain amount of correctable errors must be expected because of known physical effects.

New state:

No state change. Just the bit is set.

Proposed handling by software:

This flag can be used to analyze the state of the Flash memory. During normal operation it should be ignored. High counts of correctable errors can indicate that the Flash is operated outside of the defined operating conditions (e.g. temperature, voltage, endured p/e cycles, configured wait-states).

When programming the PFlash (end-of-line programming or SW updates) customers can count the number of correctable errors using the CBAB or rely on the PVER indication.

In case of EEPROM emulation using DFlash the verification of programmed data should be done with the normal read level and correctable errors should be ignored. Further advice can be found in **Chapter 11.8.3**.

11.8.4.2 Handling Errors During Startup

The FSR flags are not only used to inform about the success of Flash command sequences but they are also used to inform (1) the startup software and (2) the user software about special situations incurred during startup. In order to react on this information these flags must be evaluated after reset before performing any flag clearing sequence as “Clear Status” or “Reset to Read”.

The following two levels of situations are separated:

- Fatal level: the user software is not started. A WDT reset is performed.
- Warning level: the user software is started but a warning is issued.

Fatal Level (WDT Reset)

These error conditions are evaluated by the startup software which decides that the Flash is not operable and thus waits for a WDT reset. The application sees only a longer startup time followed by a WDT reset.

The reason for a failed Flash startup can be a hardware error or damaged configuration data.

Warning Level

These conditions inform the user software about an internally corrected or past error condition.

Leftover OPER:

FSR bits set: OPER.

The OPER flag are only cleared by the command sequence “Clear Status” or with a power-on reset. After any other reset a OPER flag can still be set when the user software is started.

Correctable error in UCB or configuration sector:

FSR bits set: PFSBER/PFDBER, DFSBER, DFDBER, DFTBER or ORIER.

An correctable ECC error was detected during installation of the protection or of the configuration sector content.

11.8.5 Resets During Flash Operation

A reset or power failure during an ongoing Flash operation (i.e. program or erase) must be considered as violation of stable operating conditions. However the Flash was designed to prevent damage to non-addressed Flash ranges when the reset is applied

as defined in the data sheet. The addressed Flash range is left in an undefined state. Additional means are implemented that help to prevent aborting programming processes in the DFlash.

11.8.5.1 General Advice

When an erase operation is aborted the previously programmed bits (‘1’) in the addressed Flash range can be in any state between ‘0’ and ‘1’. When reading this range all-0 can be returned, the old data, or something in between. The result can be instable. Due to the ECC correction there may even appear ‘1’ bits at positions which contained ‘0’ bits before erase start.

When a page programming operation is aborted the page can still appear as erased (but contain slightly programmed bits), it can appear as being correctly programmed (but the data has a lowered retention) or the page contains garbage data. It is also possible that the read data is instable so that depending on the operating conditions different data is read.

For the detection of an aborted Flash process the flags FSR.PROG and FSR.ERASE could be used as indicator but only when the reset was an application reset. When Flash processes are aborted by power-on resets this is not indicated by any flags. It is not possible to detect an aborted operation simply by reading the Flash range (please note that the ECC is not a reliable means to detect an aborted Flash operation). Even the margin reads don’t offer a reliable indication.

When erasing or programming the PFlash usually an external instance can notice the reset and restart the operation by erasing the Flash range and programming it again.

11.8.5.2 Advice for EEPROM Emulation

However for the case of EEPROM emulation in the DFlash this external instance is not existing. A common solution is detecting an abort by performing two operations in sequence and determine after reset from the correctness of the second the completeness of the first operation.

E.g. after erasing a DFlash sector a page is programmed. After reset the existence of correct data in this page proves that the erase process was performed completely.

The detection of aborted programming processes can be handled similarly. After programming a block of data an additional page is programmed as marker. When after reset the block of data is readable and the marker is existent it is ensured that the block of data was programmed without interruption.

In very specific cases it is allowed to repair data left from an aborted programming operation: if the algorithm can detect that an abort occurred and the algorithm knows which data must be present in the page it is possible to simply redo the programming by programming the same data again.

In the AURIX family the probability of aborting a programming process can be minimized by the following means:

- In case of a reset with stable power supply (“warm resets”) the Flash gets automatically a request to enter safe state before the reset is applied. Due to their short duration single “Write Page” operations are finished correctly by this process. “Write Burst” operations however are interrupted after the current page programming has finished.
- The voltage monitoring (see SCU chapter) can be used to get an under voltage warning early enough that an ongoing programming process can be finished and no new one is started.
- By selecting an internally generated programming voltage (see **Chapter 11.5.4.6**) the needed voltage at V_{EXT} is reduced giving more headroom for an early warning by the voltage monitors.

11.8.6 ECC

For special applications the ECC features need to be considered.

Using the PFlash Safety ECC

The PFlash Safety ECC is a vital part of the safe fetch path needed for safety applications. The difference between Legacy ECC and Safety ECC is explained in **Chapter 11.5.6.2**.

The calculation of the Safety ECC is done over 256 data bits, the address bits 22:5 and the configuration area selection signal. As side effect erased PFlash ranges or PFlash ranges programmed with the Legacy ECC will be read with ECC errors.

Thus the installation of an application using the Safety ECC should be done as follows:

- By default (delivery state) the device starts with the Safety ECC enabled.
- The Flash loader can execute from RAM. Erased PFlash ranges shall not be read.
- A sector should be programmed completely to ensure that it can be read without ECC errors.
- When changing the UCB_DFlash it is important to keep the ECC configuration flag PROCOND.NSAFECC = 0.

The installation of an application using the Legacy ECC should be done as follows:

- By default (delivery state) the device starts with the Safety ECC enabled.
- The Flash loader can execute from RAM.
- By writing FCON.NSAFECC to 1 the ECC algorithm is switched to Legacy ECC.
- Afterwards the PFlash programming is performed with the Legacy ECC.
- Erased PFlash ranges can be read without error.
- In UCB_DFlash the ECC configuration flag PROCOND.NSAFECC shall be programmed to 1 to ensure that the device starts after next reset with the Legacy ECC. It must be ensured that is kept when changing later the UCB_DFlash.

Attention: Changing FCON.NSAFECC while accessing the Flash (reading, programming, erasing) is forbidden.

Attention: Reading PFlash ranges with the incorrect ECC (e.g. Flash was programmed with Legacy ECC but reading with setting of NSAFECC=0) must not done although it might seem to work when the uncorrectable error traps are disabled.
On certain address/data combinations mis-corrections will happen and single and double-bit errors in PFlash will not be corrected because for the wrong ECC algorithm nearly all addresses will have uncorrectable errors resulting in the delivery of the raw data.

Creating Incorrect ECC

In safety applications error detection features are usually checked at each startup. In order to check the detection logic for ECC errors addresses in PFlash need to be programmed with erroneous data.

The automatic ECC generation can be disabled with ECCW.PECENCDIS. With this feature any combination of data and ECC bits can be programmed.

The ECC code causing the required error type (e.g. triple-bit error) can be determined by first programming the data with automatic ECC generation on. When reading this data the corresponding ECC code can be found in ECCRPp.RCODE. After erasing this range of Flash a modified data/ECC combination can be programmed with automatic ECC generation switched off (ECCW.PECENCDIS = 1).

It is also possible to create ECC incorrect patterns by programming twice with enabled automatic ECC generation.

Ready-to-use programming routines for the creation of ECC errors are part of the Infineon safety software “SafeTlib”.

11.8.7 Startup Tests of ECC Logic

As described in **Chapter 11.6** the ECC logic of the PFlash is protected by special hardware means, the “ECC Monitor” and the “EDC Comparator”. Depending on the targeted ASIL level these units, their error signals and generally the ECC error signaling to the SMU has to be tested during each startup after reset.

The Infineon safety software “SafeTlib” contains routines to perform these tests.

11.8.7.1 Testing ECC Alarms and Error Flags

In order to test if each type of ECC error is correctly reported to the SMU and PMU status registers a pattern with this error needs to be stored in PFlash and read during startup after reset.

Such patterns can be created as described in **Chapter 11.8.6**.

Test Pattern

As the patterns stored in PFlash might change their error signature over life-time (e.g. a single-bit error becomes a double-bit error due to a single bit toggling) each pattern needs to be stored redundantly (we propose 4 times).
As the ECC logic of each PFlash bank has separate paths the patterns have to be stored in each PFlash bank and tested in each PFlash bank.

Patterns for the following types of errors should be stored and read:

- No error (valid data). This data could be shared with the one for **Chapter 11.8.7.2**.
- 1-bit error.
- 2-bit error.
- 3-bit error to trigger an uncorrectable error.
- Address error (can be created by generating an ECC for an address that is different by 1 or 2 address bits). This test pattern is optional. It is not needed for sufficient test coverage.
- All-0 in data and ECC. This test pattern is optional. It is not needed for sufficient test coverage.
- All-1 in data and ECC. This test pattern is optional. It is not needed for sufficient test coverage.

In sum these 7 pattern (each covering one 32-byte page) are stored 4 times per PFlash bank.

For best possible robustness 2 of the 4 pattern sets shall be inverse to the other 2.

Test Flow

For each error type its 4 patterns are read. The corresponding error flag and SMU alarm should occur at least once.

11.8.7.2 Testing the “ECC Monitor”

As the ECC Monitor is always in use its logic is usually tested sufficiently during startup of the application, simply by performing read accesses to the PFlash banks.

However in order to achieve a defined error coverage a dedicated test procedure recommended.

Test Pattern

In each PFlash bank a set of 32 valid pages (i.e. containing no intentional ECC error) is stored. These patterns are chosen by IFX to ensure >90% stuck-at coverage of the ECC Monitor logic.

Test Flow

The 32 pages of each PFlash bank are read by any master e.g. the CPU.

The ECC monitor logic is working correctly if during this test the alarm “PFLASH ECC monitor error” at the SMU is not activated.

11.8.7.3 Testing the SMU Alarm of the “ECC Monitor”

In order to test that the “PFLASH ECC monitor error” alarm to the SMU can be activated the ECC monitor has to receive a pattern with incorrect ECC for which the ECC correction logic states that a correction was performed.

Test Pattern

In each PFlash bank a pattern with a correctable error has to be read, e.g. the ones used in **Chapter 11.8.7.1**.

Test Flow

The ECC correction has to be disabled for the tested PFlash bank “p” with ECCRPp.ECCORDIS. After that read 4 patterns from this bank with a correctable error. This shall trigger at least once the PFLASH ECC Error alarm in the SMU. Clear the alarm, clear ECCRPp.ECCORDIS for this bank and repeat the test for the other PFlash banks.

11.8.7.4 Testing the “EDC Comparator”

The EDC comparator compares the output of two error checking units. In case of a difference the SMU alarm “PFLASH EDC comparator error” is activated. This error can be enforced by setting for a PFlash bank “p” by setting ECCRPp.EDCERRINJ.

Test Pattern

The test patterns of **Chapter 11.8.7.1** can be used.

Test Flow

The test patterns are read with ECCRPp.EDCERRINJ = 0. This is already part of the test in **Chapter 11.8.7.1**. The “PFLASH EDC comparator error” alarm must not be activated.

The test patterns are read with ECCRPp.EDCERRINJ = 1. This shall trigger the “PFLASH EDC comparator error” alarm in the SMU.

Repeat this test for all PFlash banks.

11.8.7.5 General Advice for Startup Tests

The previously described tests need to be executed under controlled conditions:

- Only a single master with blocked interrupts should execute these tests. This is mandatory for tests with disabled ECC correction (**Chapter 11.8.7.3**) because other masters might receive uncorrected Flash data during that time.
- For the same reason the Flash test code should execute from RAM. Especially it must not execute from a PFlash bank with disabled ECC correction (**Chapter 11.8.7.3**).
- It has to be taken into account that internal pre-fetching of the PMU can trigger the “PFLASH ECC monitor error” and the “PFLASH EDC comparator error” before the corresponding Flash read access is issued by the CPU. For the ECC alarms and error flags of **Chapter 11.8.7.1** this issue is not existing as the PMU activates these errors only when the respective data is read from the bus.
- The alarms activation in the SMU has a latency of several clock cycles.
- The software executing these tests has to cope with all consequent actions like activated error flags, entries in error buffers, bus-errors, SMU alarms. Before normal execution continues these mechanisms have to be cleared.
- The existence of Flash addresses with correctable and uncorrectable errors has to be respected by the application software. Interpretation as real error has to be prevented.

12 Local Memory Unit (LMU)

The Local Memory Unit is an SRI peripheral providing access to volatile memory resources. It's primary purpose is to provide 32 kbytes of local memory for general purpose usage but it will also provide access to the separate block of emulation and debug memory (EMEM) provided in the Emulation Devices.

Data stored in the local memory is protected by ECC at all points within the LMU. Areas of local memory can be write protected by configuring up to eight address ranges using SFRs in the LMU. Each of these ranges can be sized in thirty-two byte increments and has its own, independent list of Master Tag IDs permitted write access. Read accesses are not protected.

The LMU also provides OnLine Data Acquisition (OLDA) region support. This provides an address space where writes complete without error but no memory is addressed. This allows production code to be written which writes data to memory which is only present in the Emulation Device. When running in the Emulation Device, the code maps EMEM to the OLDA region address space using the memory overlay feature and the writes are stored in the EMEM. when running in the Production Device, the LMU terminates writes to the OLDA address space without error, even though no memory exists at the target address, but the write data is discarded.

For ADAS variants of the device, the LMU acts as the interface to the hardware FFT accelerator. It allows the registers of the accelerator to be updated. and provides a data port to allow transfer of parameters.

12.1 Feature List

An overview of the features implemented in the LMU follows:

- 32 kbytes of SRAM
  - organized as 64 bit words
  - support for byte, half word and word accesses as well as double-word and burst accesses
  - memory can be used as overlay memory
- Access to the hardware FFT engine
- Protection of LMU SRAM contents
  - eight, programmable address regions can be protected
  - each address range has a programmable list of bus masters permitted write access based on the Unique master Tag ID
- Interface to the EMEM of the Emulation Device.
- Interface to the hardware FFT accelerator
- OLDA region support.

12.2 Local Memory (LMU SRAM)

The LMU SRAM can be used for code execution, data storage or overlay memory. The address range of the memory is 90000000_H to $90007FFF_H$. As well as being accessed via cached (segment 9_H), the memory can be accessed via non-cached (segment B_H) memory addresses.

The memory implements memory integrity checking for error detection and correction. This means that the memory must be initialized before reads are attempted with the integrity checking enabled to avoid generating spurious data corruption errors. Initializing before enabling the memory integrity logic allows the LMU SRAM to support initialisation using word (32 bit) or smaller writes as well as 64 bit writes. (The compiler does not support double-word memory accesses at present. Requiring double-word accesses for memory initialization requires a software workaround which it would be useful to avoid.)

If memory integrity checking is enabled, a read access which fails the integrity check will be terminated with an SRI error condition. This behavior can be changed by setting the **LMU_MEMCON.ERRDIS** bit to 1_B. If the bit is set then an SRI error will not occur.

An ECC error will also be reported to the SMU. The SMU will use this signal for error indication and triggering of an NMI trap (if enabled).

The LMU SRAM is internally organized as a 64 bit memory without the possibility of sub-word accesses. This means that any write access of less than 64 bits of data needs an internal Read-Modify-Write (iRMW) operation to correctly write the data and update the ECC data. This happens transparently to rest of the system unless an uncorrected ECC error is detected during the read phase of the iRMW. This ECC error will be flagged in the same way as a data ECC error occurring during a normal read. In addition the **LMU_MEMCON.RMWERR** flag will be set. The write operation will not take place as this would write incorrect ECC data to the memory (the ECC would match the written data but the data would potentially contain an uncorrected error).

LMU SRAM performance will be the same as, or better than, the performance of the embedded flash. This applies to both the initial latency of the first word returned and also the incremental latency for each word in the same cache line fetched.

If the CPU access can't be handled by the LMU SRAM (e.g. an unsupported SRI opcode has been received), an SRI bus error is reported by the LMU. This will cause a DSE trap.

Some bitfields of the **LMU_MEMCON** register are protected by **LMU_MEMCON.PMIC** bit. If the data written to the register has the bitfield set to 0_B, no change will be made to bits 15_D to 9_D of the register regardless of the data written to these fields.

12.2.1 LMU SRAM Read Buffers

The LMU SRAM interface implements read buffers to optimize use of the LMU SRAM. The read buffer will store the last 64 bit word accessed from the LMU SRAM. The buffer is predominantly useful when sequential, 32 bit accesses are being performed.

One read buffer will be available for every processor instance in the system. Each buffer will be controlled by an instance of the **LMU_BUFCONx (x=0-2)** register. The read buffer can be enabled using the **LMU_BUFCONx (x=0-2).TAG1** or **LMU_BUFCONx (x=0-2).TAG2** bits. Setting either of the bits to 1_B will enable the associated buffer. Each buffer can be associated with one or two Unique Master Tag IDs (**LMU_BUFCONx (x=0-2).TAG1** and **LMU_BUFCONx (x=0-2).TAG2**) and an update of the buffer will only be triggered when an access to the LMU SRAM uses an enabled tag. The tags are independently enabled using two control bits, **LMU_BUFCONx (x=0-2).EN1** for **TAG1** and **LMU_BUFCONx (x=0-2).EN2** for **TAG2**.[1)]

12.3 Memory Protection

The LMU allows for the definition of eight, protected regions of SRAM memory. The protection applies only to write accesses to SRAM included in the LMU (including atomic read-modify-write operations), not EMEM or registers. SRAM read accesses are not protected.

The protection scheme is based on the use of Unique Master Tag IDs to identify the master attempting the access and allows for a six bit tag individually identifying up to 64 masters. 32 masters are supported in the current implementation.

Each region is defined using four registers, **LMU_RGNLAx (x=0-7)** to define the lower address of the region, **LMU_RGNUAx (x=0-7)** to define the upper address of the region and **LMU_RGNACCENAx (x=0-7)** and **LMU_RGNACCENBx (x=0-7)** to individually select the master tags permitted write access to the defined address range. The scheme is compatible with the Tricore implementation so **LMU_RGNLAx (x=0-7)** defines the first address in the region

After reset, the region address registers will be set to include the whole of the LMU, SRAM address space and write access by all masters will be enabled.

The registers implementing the memory protection scheme are protected by the “safety endinit” function.

If overlapping regions are defined, then a write access only needs to be permitted by one of the overlapping regions for it to succeed.

Legal addresses for write accesses are those to a defined address region with a permitted master tag. The memory protection scheme will block write accesses to an address falling outside all of the defined address ranges

When altering protection settings, it should be noted that, due to access pipelining in the LMU and resynchronization delays in the register block, a write to a memory address affected by the protection change occurring immediately after the register write initiating the change may, or may not, be affected by the changed settings.

1) Multiple buffers should not be configured to respond to the same Master Tag ID. LMU behaviour in this case is undefined.

12.4 FFT Accelerator Interface

The LMU provides maps the resources of the hardware based FFT accelerator into the system address space. For full information on using the FFT accelerator, see the dedicated chapter of this specification.

Accesses to the registers and the data load/unload ports of the FFT accelerator are handled separately.

The LMU provides two functions related to the data load/unload ports:

- Any access to the data load/unload ports which is not completed by the FFT accelerator will time out after 255 clock cycles
- Read accesses to the data unload port are augmented by a prefetch function. Any read access using a BTR4, SSI opcode will automatically trigger another access of the same size to the next contiguous address. The prefetch is enabled by writing bit 10 (FFTPFT) of the MEMCON register.

12.5 Emulation Memory (EMEM)

In the Emulation Device , an area of Emulation Memory (EMEM) is provided, which can be used for either calibration via overlay of non-volatile memory or overlay of the OLDA address space (see **Chapter 1.8** below). The address range allocated for the memory is $9F000000_H$ to $9F3FFFFF_H$ which allows the maximum EMEM size fitted in any of the derivative products to be addressed.

As well as the cached addresses (segment 9_H), noncached address (segment B_H) accesses can be used for EMEM accesses via the LMU.

The Emulation Memory interface controls the CPU-accesses to the Emulation Memory in the Emulation Device. All widths of write accesses are supported (byte, halfword, word, double-word).

CPU-controlled Load-Modify-Store accesses (with LDMST instruction) are supported as separate read and write instructions not as an atomic operation.

In the production device, the EMEM interface is always disabled. A CPU read access from the Emulation Memory region causes a DSE trap by returning an SRI bus error. If the Emulation Memory region read access is initiated by a SPB master (e.g. PCP), additionally a SPB error interrupt can be generated.

By default, write accesses to the Emulation Memory by any master will cause an SRI bus error trap in the production device.

In the Emulation Device, a SRI bus error is returned by the LMU if a read access can’t be handled by the EMEM, for example, when the CPU accesses a trace memory tile in EMEM. In this case, the EMEM access is aborted by the LMU.

Write accesses which cannot be handled by the EMEM will fail silently as the write access is completed on the SRI bus before being passed to the EMEM. Therefore any error condition encountered by the EMEM will occur after the SRI access has completed.

A status bit **LMU_MEMCON.EWERR** will be set if an error occurs on an EMEM write. This bit can be cleared by writing 0_B.

Wait states can be manually added to EMEM memory accesses using the **LMU_MEMCON.WSTATES** field. This can be programmed with a counter preload value. A counter is loaded with this value every time the LMU starts a read access to the EMEM. Data returned by the EMEM will be delayed until the counter reaches a count of zero. The counter is decremented by one every SRI clock cycle.

12.5.1 EMEM Memory Read Buffers

Accesses to the EMEM address space can be stored to read buffers. One read buffer will be instantiated for each CPU in the system. Each of these buffers will store 256 bits of data. Buffering of read data is controlled by the **LMU_BUFCONx (x=0-2).EREN** bit and will be enabled when **EREN** is 1_B. Speculative prefetch is available only if buffering of read data is enabled and is controlled by the **LMU_BUFCONx (x=0-2).EPEN** bit. Setting **EPEN** to 1_B will allow prefetch if **EREN** is also 1_B. Two Unique Master Tag IDs can be manually associated with each CPU using the **LMU_BUFCONx (x=0-2).TAG1** and **LMU_BUFCONx (x=0-2).TAG2**. Each tag field can be individually enabled using the **LMU_BUFCONx (x=0-2).EN1** and **LMU_BUFCONx (x=0-2).EN2** fields respectively.[1)]

If read buffering is enabled, then all reads to the EMEM not using the BTR4 opcode will be mapped to BTR4 equivalent read accesses to the EMEM and the data returned stored in the related buffer while the requested data is returned to the SRI interface. Any further reads to the address range of the data stored in the buffer will be returned directly from the buffer.

If prefetch is enabled, then two conditions will trigger a prefetch:

- any SDTW read to the EMEM will cause a speculative read of the next BTR4 address range to be triggered if the address of the read corresponds to the highest 32 bit address of a BTR4[2)] (even if the data for the read is already stored in the buffer).
- Any BTR4 read to the cached segment address of the EMEM (9_H) will trigger a prefetch (even if the data for the read is already stored in the buffer).

The prefetched data will be stored in the related buffer and any reads to the address range of the prefetch will be returned directly from the buffer. A prefetch will not occur if another read to EMEM is already required. Write accesses will be scheduled after the prefetch has completed.

Prefetch takes priority over read buffering.

1) Multiple buffers should not be configured to respond to the same Master Tag ID. LMU behaviour in this case is undefined.

2) A BTR4 access requires 256 bits of data or 8, 32 bit words. These words always have $A_{SRI}[4:2] = 000_B$, 001_B, 010_B, 011_B, 100_B, 101_B, 110_B, 111_B. An 32 bit access with $A_{SRI}[4:2] = 111_B$ will trigger a prefetch

Any write access to EMEM which collides with the address range stored in a buffer will invalidate that buffer. Read accesses to the Emulation Device register space will not be buffered.

12.5.2 Access to Emulation Device Register Space

In addition to accessing EMEM, the EMEM interface in the LMU is also used to access configuration registers in the Emulation Device. To allow this accesses to the address range $F9000000_H$ to $F90FFFFF_H$ are directed to the ED register address space via the EMEM interface.

The interface to the ED register space only supports 32 bit accesses. Any other size of access will be errored.

12.6 Error Detection and Signalling

The LMU will detect several different classes of error which cannot be signalled on the SRI during the associated transaction. These will cause a flag to be set in the **LMU_MEMCON** register and a trigger will be sent to either the SMU or the Interrupt system for processing. In general ECC errors will be processed by the SMU and other errors will be passed to the Interrupt Router. Unless explicitly stated below, the access will complete. The following list details the detected error conditions:

12.6.1 EMEM Read Error

An error signalled by the EMEM on the second data phase of a 256 bit access cannot be signalled on the SRI as this would violate the protocol or, in the case of a 32 bit read, occur after the SRI access has completed. In this event, the LMU_MEMCON.ERERR bit will set and a trigger will be sent to the Interrupt Router.

12.6.2 EMEM Write Error

All errors signalled by the EMEM during a write access will, by definition in the SRI protocol, occur after the access has been accepted from the SRI. These errors will set the LMU_MEMCON.EWERR bit and send a trigger to the Interrupt router

12.6.3 Internal ECC Error

Internal registers used to transfer data between the RAM and the SRI interface are ECC protected. In the event of this ECC detecting an error, the LMU_MEMCON.INTERR bit will be set and an error will be signalled to the SMU.

12.6.4 Internal SRAM Read Error

The LMU will perform a internal Read-Modify-Write (iRMW) access when a write of less than 64 bits of data is performed. An ECC error reported by the RAM on the read phase

will cause the LMU_MEMCON.RMWERR bit to be set and an error will be signalled to the SMU. The write phase of the iRMW will not take place.

12.6.5 ECC check failure

The hardware used to check and correct the read data from the SRAM is replicated and the output from the two instances is compared. In the event of a difference between the two outputs, an error condition will be signalled to the SMU. The second instance will use inverted logic to eliminate common failure modes.

12.6.6 SRI write access data phase error

If an ECC error occurs on the data phase of an SRI write access then the LMU_MEMCON.DATAERR bit will be set and an error will be signalled to the SMU

12.6.7 SRI access address phase error

If an ECC error occurs on the address phase of an SRI access then the LMU_MEMCONADDERR bit will be set and an error will be signalled to the SMU. The SRI access will terminate with an error.

12.7 Online Data Acquisition (OLDA) and its Overlay

Calibration is additionally supported by an OLDA memory range of up to 32 Kbyte, which is a virtual memory and physically only available if it is redirected (by the overlay feature of the processor) to internal or external physical memory or to the EMEM in the Emulation Device.

If OLDA support is enabled in the LMU, direct write accesses (without redirection) to the OLDA range are not really executed, and they do not generate a bus error trap[1)]. If OLDA support is not enabled, write accesses will generate a bus error trap. OLDA support is enabled by setting LMU_MEMCON.OLDAEN to 1_B.

Note that switching on or off the OLDA function takes a finite amount of time after the register write completes. It is therefore possible for an immediately following access pipelined into the LMU SRI interface to be acknowledged with either the "on" or "off" behaviour.

Read accesses to the OLDA range generate a bus error trap, if not redirected to a physically available overlay block. Successful accesses to the OLDA memory range will only take place when the accesses are redirected to real, physical memory.

1) Write accesses to a cached memory address will trigger a read to fill the cache line before the data is written to the cache. This read will trigger a bus error.

The base address of the virtual OLDA memory range is A/8FE7 0000_H, the end address is A/8FE7 $7FFF_H$. Accesses to the OLDA range are also supported in cached address space.

Note: In OTARx registers, any target address can be selected for redirection, thus also addresses in the OLDA range. However, the handling of direct accesses to the OLDA range is completely controlled in the LMU.

12.8 Clock Control

The LMU contains a clock control register, **LMU_CLC**, which allows the LMU to be put into a power saving mode.

If LMU_CLC.DISR is set then the LMU will be disabled and all accesses will be errored unless they are addressed to a register.

12.9 LMU Register Protection

The LMU implements the standard memory protection scheme for peripheral registers using the **LMU_ACCEN0** and **LMU_ACCEN1** registers. This allows the LMU control registers to be protected from corruption by untrusted masters. Masters are identified using the SRI tag of the access and, if the appropriate bit is not set in the access enable registers, write accesses will be disconnected with error acknowledge. See the On Chip Bus chapter for the product's master Tag ID to master peripheral mapping. This protection scheme does not apply to the **LMU_ACCEN0** and **LMU_ACCEN1** themselves.

LMU_ACCEN0 and **LMU_ACCEN1** are protected by Safe Endinit while all other registers are Endinit protected. The Endinit and Safe Endinit system status is defined by different Watchdog Units in the System Control Unit (SCU).

12.10 LMU Registers

The LMU registers are mapped into a 256 byte address space which allows for 64 registers. Accesses to unused register space will cause an SRI bus error.

Table 12-1 Registers Address Space

Module	Base Address	End Address	Note
LMU	F870 0800$_H$	F870 08FF$_H$	All registers are endinit or safe endinit protected and are accessible in Supervisor mode only

Table 12-2 Registers Overview

Short Name	Description	Offset Addr.[1)]	Access Mode		Reset Class	Page Number
			Read	Write		
LMU_CLC	LMU Clock Control	0$_H$	SV,32	SV,E,P,32	3	4-33
RESERVED	reserved address space	4$_H$	BE	BE	n.a.	
LMU_MODID	LMU Module ID	8$_H$	SV,32	R	3	4-34
RESERVED	reserved address space	C$_H$	BE	BE	n.a.	
LMU_ACCEN0	LMU Access Enable 0	10$_H$	SV,32	SV,SE,32	3	4-35
LMU_ACCEN1	LMU Access Enable 1	14$_H$	SV,32	SV,SE,32	3	4-36
RESERVED	reserved address space	18$_H$ to 1C$_H$	BE	BE	n.a.	
LMU_MEMCON	LMU Memory Control	20$_H$	SV,32	SV,E,P,32	3	4-37
RESERVED	reserved address space	24$_H$ to 2C$_H$	BE	BE	n.a.	
LMU_BUFCON 0	LMU Buffer Control	30$_H$	SV,32	SV,E,P,32	3	4-41
LMU_BUFCON 1	LMU Buffer Control	34$_H$	SV,32	SV,E,P,32	3	4-41

Table 12-2 Registers Overview (cont'd)

Short Name	Description	Offset Addr.[1)]	Access Mode		Reset Class	Page Number
			Read	Write		
LMU_BUFCON2	LMU Buffer Control	38_H	SV,32	SV,E,P,32	3	4-41
RESERVED	reserved address space	$3C_H$ to $4C_H$	BE	BE	n.a.	
LMU_RGNLA0	LMU Region Lower Address	50_H	SV,32	SV,SE,P,32	3	4-43
LMU_RGNUA0	LMU Region Upper Address	54_H	SV,32	SV,SE,P,32	3	4-44
LMU_RGNACCENA0	LMU Region Access Enable A	58_H	SV,32	SV,SE,P,32	3	4-45
LMU_RGNACCENB0	LMU Region Access Enable B	$5C_H$	SV,32	SV,SE,P,32	3	4-46
LMU_RGNLA1	LMU Region Lower Address	60_H	SV,32	SV,SE,P,32	3	4-43
LMU_RGNUA1	LMU Region Upper Address	64_H	SV,32	SV,SE,P,32	3	4-44
LMU_RGNACCENA1	LMU Region Access Enable A	68_H	SV,32	SV,SE,P,32	3	4-45
LMU_RGNACCENB1	LMU Region Access Enable B	$6C_H$	SV,32	SV,SE,P,32	3	4-46
LMU_RGNLA2	LMU Region Lower Address	70_H	SV,32	SV,SE,P,32	3	4-43
LMU_RGNUA2	LMU Region Upper Address	74_H	SV,32	SV,SE,P,32	3	4-44
LMU_RGNACCENA2	LMU Region Access Enable A	78_H	SV,32	SV,SE,P,32	3	4-45
LMU_RGNACCENB2	LMU Region Access Enable B	$7C_H$	SV,32	SV,SE,P,32	3	4-46
LMU_RGNLA3	LMU Region Lower Address	80_H	SV,32	SV,SE,P,32	3	4-43
LMU_RGNUA3	LMU Region Upper Address	84_H	SV,32	SV,SE,P,32	3	4-44

Table 12-2 Registers Overview (cont'd)

Short Name	Description	Offset Addr.[1)]	Access Mode		Reset Class	Page Number
			Read	Write		
LMU_RGNACCENA3	LMU Region Access Enable A	88_H	SV,32	SV,SE,P,32	3	4-45
LMU_RGNACCENB3	LMU Region Access Enable B	$8C_H$	SV,32	SV,SE,P,32	3	4-46
LMU_RGNLA4	LMU Region Lower Address	90_H	SV,32	SV,SE,P,32	3	4-43
LMU_RGNUA4	LMU Region Upper Address	94_H	SV,32	SV,SE,P,32	3	4-44
LMU_RGNACCENA4	LMU Region Access Enable A	98_H	SV,32	SV,SE,P,32	3	4-45
LMU_RGNACCENB4	LMU Region Access Enable B	$9C_H$	SV,32	SV,SE,P,32	3	4-46
LMU_RGNLA5	LMU Region Lower Address	$A0_H$	SV,32	SV,SE,P,32	3	4-43
LMU_RGNUA5	LMU Region Upper Address	$A4_H$	SV,32	SV,SE,P,32	3	4-44
LMU_RGNACCENA4	LMU Region Access Enable A	$A8_H$	SV,32	SV,SE,P,32	3	4-45
LMU_RGNACCENB5	LMU Region Access Enable B	AC_H	SV,32	SV,SE,P,32	3	4-46
LMU_RGNLA6	LMU Region Lower Address	$B0_H$	SV,32	SV,SE,P,32	3	4-43
LMU_RGNUA6	LMU Region Upper Address	$B4_H$	SV,32	SV,SE,P,32	3	4-44
LMU_RGNACCENA6	LMU Region Access Enable A	$B8_H$	SV,P,32	SV,SE,P,32	3	4-45
LMU_RGNACCENB6	LMU Region Access Enable B	BC_H	SV,32	SV,SE,P,32	3	4-46
LMU_RGNLA7	LMU Region Lower Address	$C0_H$	SV,32	SV,SE,P,32	3	4-43
LMU_RGNUA7	LMU Region Upper Address	$C4_H$	SV,32	SV,SE,P,32	3	4-44

Table 12-2 Registers Overview (cont'd)

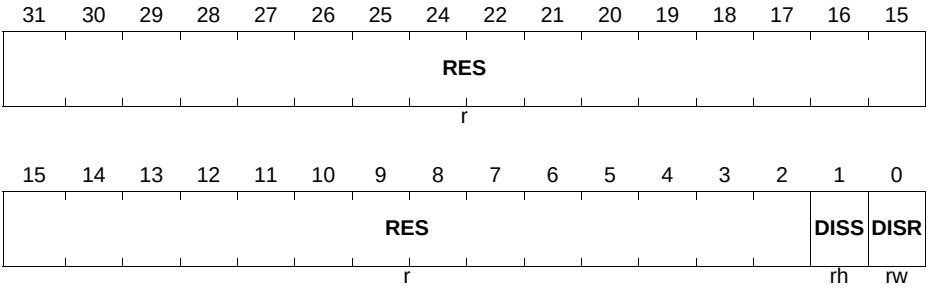
Short Name	Description	Offset Addr.[1]	Access Mode		Reset Class	Page Number
			Read	Write		
LMU_RGNACCENA7	LMU Region Access Enable A	$C8_H$	SV,32	SV,SE,P,32	3	4-45
LMU_RGNACCENB7	LMU Region Access Enable B	CC_H	SV,32	SV,SE,P,32	3	4-46
RESERVED	reserved address space	$D0_H$ to FF_H	BE	BE	n.a.	

1) The absolute register address is calculated as follows:
Module Base Address (**Table 1-5**) + Offset Address (shown in this column)

LMU Clock Control Register

LMU_CLC

LMU Clock Control Register **(000_H)** **Reset Value: 0000 0000_H**

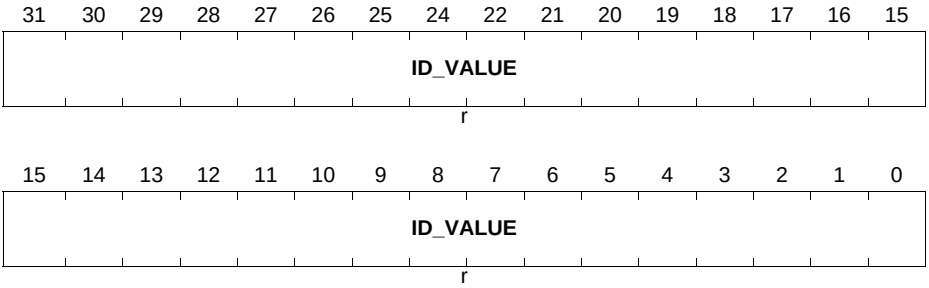


Field	Bits	Type	Description
DISR	0	rw	**LMU Disable Request Bit** This bit is used for enable/disable control of the LMU. 0_B LMU disable is not requested 1_B LMU disable is requested
DISS	1	rh	**LMU Disable Status Bit** Current state of LMU. 0_B LMU is enabled (default after reset) 1_B LMU is disabled
RES	31:2	r	**Reserved**

LMU Module ID Register

LMU_MODID

LMU Module ID Register **(008_H)** **Reset Value: $0088\ C002_H$**



Field	Bits	Type	Description
ID_VALUE	31:0	r	**Module Identification Value**

LMU Access Enable Register 0

The Access Enable Register 0 controls write access for transactions to registers with the on chip bus master TAG ID 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The LMU is prepared for an 6 bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6 bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000_B, EN1 -> TAG ID 000001_B, ... ,EN31 -> TAG ID 011111_B.

LMU_ACCEN0
LMU Access Enable Register 0 (10_H) Reset Value: FFFF FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN 31	**EN 30**	**EN 29**	**EN 28**	**EN 27**	**EN 26**	**EN 25**	**EN 24**	**EN 23**	**EN 22**	**EN 21**	**EN 20**	**EN 19**	**EN 18**	**EN 17**	**EN 16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN 15	**EN 14**	**EN 13**	**EN 12**	**EN 11**	**EN 10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

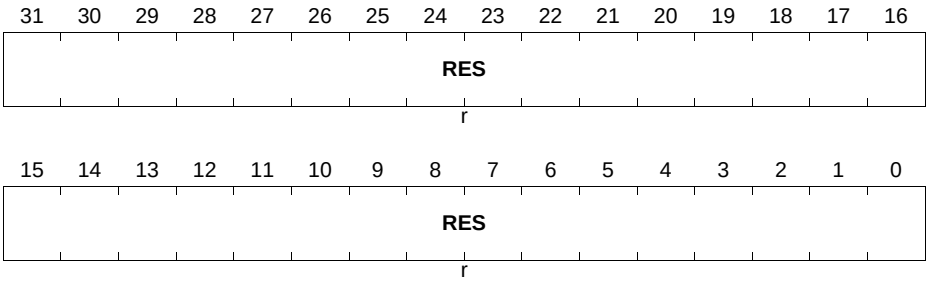
Field	Bits	Type	Description
ENn (n = 0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the LMU register addresses for transactions with the Master TAG ID n 0_B Write access will not be executed. Read accesses will be executed. 1_B Write and read accesses will be executed

LMU Access Enable Register 1

The Access Enable Register 1 controls write access for transactions with the on chip bus master TAG ID 100000_B to 111111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). These tags are not used in this system and so no programmable bits are provided.

LMU_ACCEN1

LMU Access Enable Register 1 (14_H) Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
RES	31:0	r	**Reserved** Read as 0; should be written with 0.

LMU Memory Control Register

Provides Control of the memory integrity error checking, error signalling to the SMU and error injection for ECC logic test. Also control of the OLDA function. The register is cleared by an application reset.

LMU_MEMCON
LMU Memory Control Register (020$_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WSTATES				0	FFT PFT	ERR DIS	PMI C	ADD ERR	DAT AER R	EWE RR	RM WER R	ERE RR	INTE RR	POL DAE N	OLD AEN
rw				r	rw	rw	w	rwh	rwh	rwh	rwh	rwh	rwh	w	rw

Field	Bits	Type	Description
OLDAEN	0	rw	**Online Data Acquisition Enabled** This bit is used to control trap generation for write accesses to the OLDA address range. 0_B Trap generation on write access to OLDA memory range is enabled 1_B No trap generated on write access to OLDA memory range.
POLDAEN	1	w	**Protection Bit for OLDAEN** This bit always returns 0_B when read. 0_B Bit Protection: Bit OLDAEN remains unchanged after LMU_MEMCON write. 1_B OLDAEN can be changed by current write to LMU_MEMCON
INTERR	2	rwh	**Internal ECC Error** Flag set by hardware when the LMU logic detects an ECC error while accessing the RAM. This bit is cleared by writing 0_B but cannot be set by software. 0_B No error has occurred 1_B An error has been observed during a RAM access.

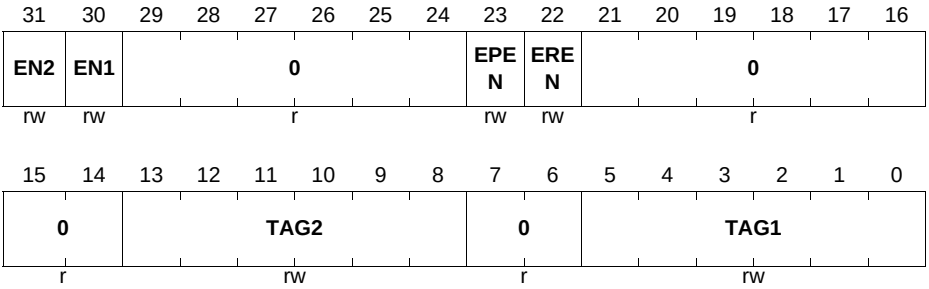
Field	Bits	Type	Description
ERERR	3	rwh	**EMEM Read Error** Flag set by hardware when the LMU logic detects an error on an EMEM read transaction which cannot be reported on the SRI bus. This bit is cleared by writing 0_B but cannot be set by software. 0_B No error has occurred 1_B An unreportable error has been observed on an EMEM read transaction.
RMWERR	4	rwh	**Internal Read Modify Write Error** Flag set by hardware when the LMU logic detects an ECC error on the read phase of an internal RMW operation. This bit is cleared by writing 0_B but cannot be set by software. 0_B No error has occurred 1_B An error has been observed during an iRMW operation.
EWERR	5	rwh	**EMEM Write Error** Flag set by hardware when the LMU logic detects an error on an EMEM write transaction. This bit is cleared by writing 0_B but cannot be set by software. 0_B No error has occurred 1_B An error has been observed on an EMEM write transaction.
DATAERR	6	rwh	**SRI Data Phase ECC Error** Flag set by hardware when the SRI interface detects an ECC error in the data phase of an incoming write transaction. This bit is cleared by writing 0_B but cannot be set by software. 0_B No error has occurred 1_B An ECC error has been observed on an SRI transaction addressed to the LMU
ADDERR	7	rwh	**SRI Address Phase ECC Error** Flag set by hardware when the SRI interface detects an ECC error in the address phase of an incoming transaction. This bit is cleared by writing 0_B but cannot be set by software. 0_B No error has occurred 1_B An ECC error has been observed on an SRI transaction addressed to the LMU

Field	Bits	Type	Description
PMIC	8	w	**Protection Bit for Memory Integrity Control Bit** Will always return 0_B when read 0_B Bit Protection: Bit 9 remains unchanged after LMU_MEMCON write. 1_B Bit 9 will be updated by the current write to LMU_MEMCON
ERRDIS	9	rw	**ECC Error Disable** When set SRI bus errors caused by ECC errors in data read from the SRAM will be disabled 0_B Normal behavior. SRI error will occur on SRAM ECC errors. Default after reset 1_B Test Mode. SRI errors will not be generated on an SRAM ECC error. This does not affect the generation of interrupts.
FFTPFT	10	rw	**FFT Accelerator Prefetch Disable** 0_B Read accesses to FFT accelerator data port using a BTR4, SRI opcode trigger prefetch of the next data. 1_B No prefetch on accesses to FFT accelerator data port
0	11	r	**Reserved** Read as 0_H, must be written as 0_H
WSTATES	15:12	rw	**EMEM Wait States** Wait State Counter for EMEM accesses. See **“Emulation Memory (EMEM)” on Page 4-6** for operational details.
0	31:16	r	**Reserved** Read as 0_H, must be written as 0_H

LMU Buffer Control Register

Enables control of a read buffer. The register is cleared by an application reset.

LMU_BUFCONx (x=0-2)
LMU Buffer Control Register **(030_H+x*04_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
TAG1	5:0	rw	**Master Tag ID 1** This field should store the Master Tag ID of accesses which are required to trigger an update of the read and prefetch buffers.
0	7:6	r	**Reserved** Read as 0_B, should be written with 0_B.
TAG2	13:8	rw	**Master Tag ID 2** This field should store the Master Tag ID of accesses which are required to trigger an update of the read and prefetch buffers.
0	21:14	r	**Reserved** Read as 0_B, should be written with 0_B.
EREN	22	rw	**EMEM Read Buffer Enable** Enable bit for the read buffer function on EMEM reads 0_B Do not buffer reads from the EMEM 1_B Read buffer is enabled.
EPEN	23	rw	**EMEM Prefetch Enable** Enable bit for the prefetch function on EMEM reads 0_B Do not prefetch from the EMEM 1_B Prefetch is enabled.

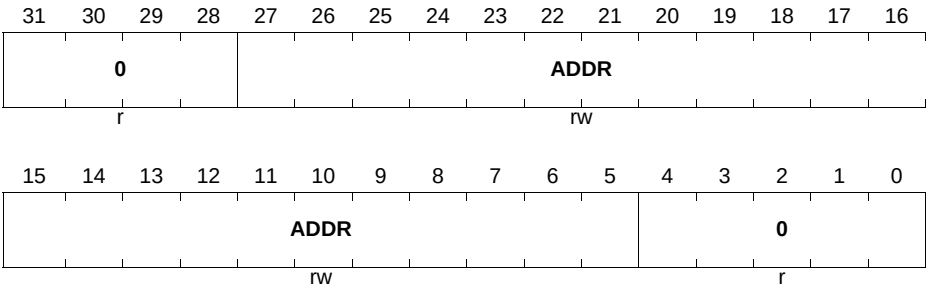
Field	Bits	Type	Description
0	29:24	r	**Reserved** Read as 0_B, should be written with 0_B.
EN1	30	rw	**TAG1 Field Enable** Enable bit for the TAG1 field 0_B TAG1 does not contain a valid Unique Master Tag ID value and should be ignored. 1_B TAG1 does contain a valid Unique Master Tag ID value and should be used when triggering an update for the read buffer. Local SRAM Read buffer is enabled.
EN2	31	rw	**TAG2 Field Enable** Enable bit for the TAG1 field 0_B TAG2 does not contain a valid Unique Master Tag ID value and should be ignored. 1_B TAG2 does contain a valid Unique Master Tag ID value and should be used when triggering an update for the read buffer. Local SRAM Read buffer is enabled.

LMU Region Lower Address Register

In conjunction with the associated RGNUA and RGNACCENx registers, the RGNLA register provides control of a memory protection region. The register is cleared by a application (Class 3) reset. RGNLA defines the lower address of a region of memory, RGNUA defines the upper address and the RGNACCENx registers define the Unique Master Tag IDs allowed to access the region. Address ranges can bet set to be larger than the LMU address space but only accesses to the LMU are affected by these registers. Bits 31_D to 28_D are ignored as they define the segment address and the minimum resolution of the comparison logic is 32_D bytes so address bits 4_D down to 0_D are similarly not used. Bit 28 of RGNUA.ADDR can be set if all addresses in the segment greater than or equal to RGNLA.ADDR are to be considered valid

LMU_RGNLAx (x=0-7)

LMU Region Lower Address Register(050_H+x*10_H) **Reset Value: 0000 0000_H**



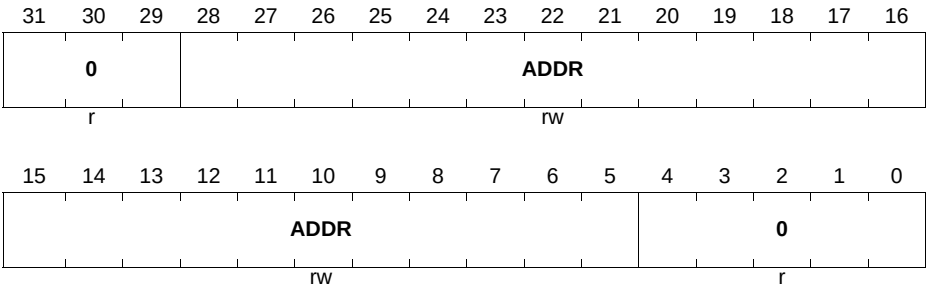
Field	Bits	Type	Description
0	4:0	r	**Unused** Read as 0_B, should be written with 0_B although written value will be ignored.
ADDR	27:5	rw	**Region Lower Address** Bits 27 to 5 of the SRI address which is the lower bound of the defined memory region
0	31:28	r	**Unused** Read as 0_B, should be written with 0_B although written value will be ignored.

LMU Region Upper Address Register

In conjunction with the associated RGNLA and RGNACCENx registers, the RGNUA register provides control of a memory protection region. The register is cleared by an application (Class 3) reset. RGNUA defines the upper address of the memory region and will contain the first address outside the protected region.

LMU_RGNUAx (x=0-7)

LMU Region Upper Address Register(054_H+x*10_H) **Reset Value: 1000 0000_H**



Field	Bits	Type	Description
1	4:0	r	**Unused** Read as 0_B, should be written with 0_B although written value will be ignored.
ADDR	28:5	rw	**Region Lower Address** Bits 27 to 5 of the SRI address which is the upper bound of the defined memory region. i.e. the first address outside the protected region. Set bit 28 if all addresses in the segment greater than the lower address are to be accepted
0	31:28	r	**Unused** Read as 0_B, should be written with 0_B although written value will be ignored.

LMU Region Access Enable Register A

The Access Enable Register A controls write access for transactions to the protected memory region with the on chip bus master TAG ID 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The LMU is prepared for an 6 bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6 bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000_B, EN1 -> TAG ID 000001_B, ... ,EN31 -> TAG ID 011111_B.

LMU_RGNACCENAx (x=0-7)

LMU Region Access Enable Register 0(058_H+x*10_H) **Reset Value: FFFF FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN 31	**EN 30**	**EN 29**	**EN 28**	**EN 27**	**EN 26**	**EN 25**	**EN 24**	**EN 23**	**EN 22**	**EN 21**	**EN 20**	**EN 19**	**EN 18**	**EN 17**	**EN 16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN 15	**EN 14**	**EN 13**	**EN 12**	**EN 11**	**EN 10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n = 0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write accesses not permitted for this region 1_B Read and Write permitted for this region

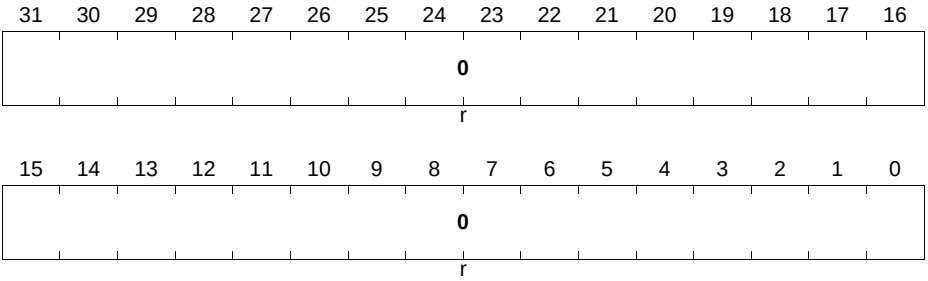
LMU Region Access Enable Register B

The Access Enable Register B controls write access for transactions to the protected memory region with the on chip bus master TAG ID 100000_B to 111111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The LMU is prepared for an 6 bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6 bit TAG ID encoding.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000_B, EN1 -> TAG ID 100001_B, ... ,EN31 -> TAG ID 111111_B.

LMU_RGNACCENBx (x=0-7)

LMU Region Access Enable Register 1($05C_H$+x*10_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
0	31:0	r	**Reserved** Read as 0; should be written with 0.

13 Data Access Overlay (OVC)

The data overlay provides the capability to redirect selected data accesses to the Overlay memory. Data accesses made by the TriCore to Program Flash, Online Data Acquisition space, or EBU space can be redirected. Overlay memory may be located in the Local Memory (if present), in the Emulation Memory (Emulation Device only), or in the DPSR/PSPR memory. Overlay functionality makes it possible, for example, to modify the application's test and calibration parameters (which are typically stored in Flash memory) during run time of a program. Note that only read and write data accesses are redirected. The access redirection is executed without performance penalty.

Attention: As the address translation is implemented in the DMI it is only effective for data accesses by the TriCore. Instruction fetches by the TriCore or accesses by any other master (including the debug interface) are not effected!

Summary of Features and Functions

- Redirecting data accesses addressed to Program Flash, OLDA or External EBU Space.
- Support redirection to Overlay memory located in:
  - Local Memory (LMU) (if present)
  - Emulation Memory (Emulation Device only)
  - DSPR or PSPR memory
- Support of up to 4 MB overlay memory address range;
- Up to 8 overlay ranges ("blocks") available;
- Overlay block size from 32 byte to 128 Kbyte;
- Up to 1 MB of redirected space (8 ranges x 128 Kbyte);
- Overlay memory location and block size selected individually for every overlay block;
- Multiple overlay blocks can be enabled or disabled with one register write access;
- Programmable flush (invalidate) control for data cache in DMI.
- Overlay start/stop synchronised against data load.

13.1 Data Access Redirection

The principle of redirecting data access from the original target memory (“Target Address”) to overlay memory (“Redirected Address”) is shown below.

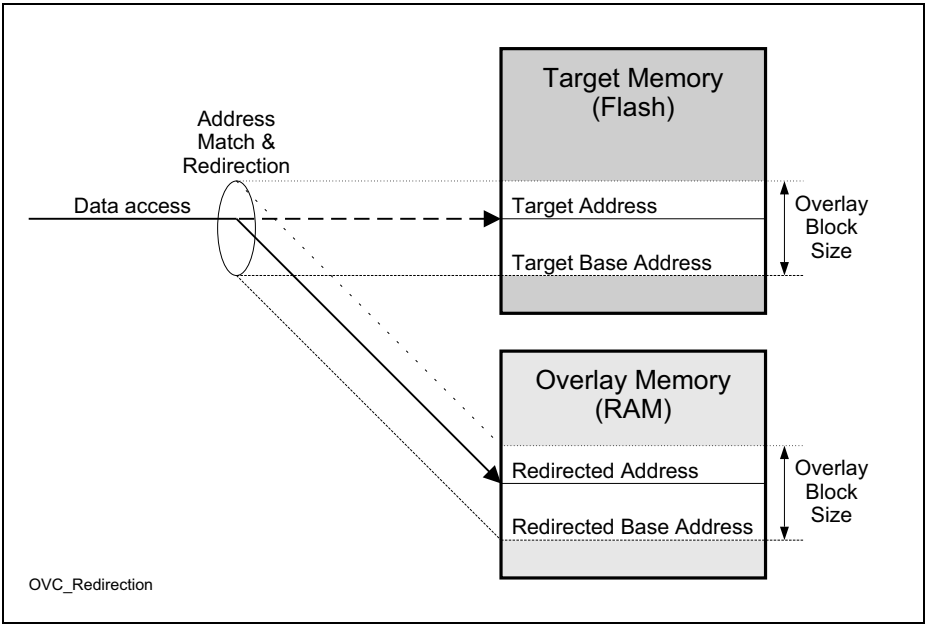


Figure 13-1 Data Access Redirection

Data access overlay is defined using overlay ranges (“overlay blocks”). Each overlay block defines one continuous range of address space for which the accesses are redirected. Each overlay block is configured with the following parameters:

- Overlay Block Target Base Address - the start address for the range of the target addresses to be redirected;
- Overlay Block Size - the size of the range of the addresses to be redirected;
- Overlay Block Redirection Base Address - the start address for redirection.

In TC21x/TC22x/TC23x up to 8 overlay ranges can be used in each TriCore instance.

Each overlay block has 3 associated registers for independent configuration of these parameters. The overlay parameters are configured as follows:

- Target Base Address is configured with OTARx register (**“OTARx (x=0-7)” on Page 13-13**),
- Overlay Block Size is configured with OMASKx register (**“OMASKx (x=0-7)” on Page 13-14**),

- Redirection Base Address is configured with RABRx register (**"RABRx (x=0-7)" on Page 13-11**).

The size of the overlay memory blocks can be 2^n x 32 bytes, with n = 0 to 12. This gives the block size range from 32 bytes to 128 Kbytes. The start address of the block can only be an integer multiple of the programmed block size (natural alignment boundary). If OTAR register value, or RABR register value is not aligned with the block size, the least significant bit values are ignored and treated as zero.

The Redirection Base Address is determined by two fields in RABRx register:

- RABRx.OMEM selecting the Overlay Memory, and
- RABRx.OBASE selecting the base address within this memory.

Each overlay block can be activated or deactivated with its RABRx.OVEN bit. Overlay blocks can be activated and deactivated independently, by directly accessing RABRx register, or in groups, where multiple configured blocks are activated of deactivated concurrently. For information on concurrent block activation see **Chapter13.4.1**.

The address redirection process is shown in the following figure.

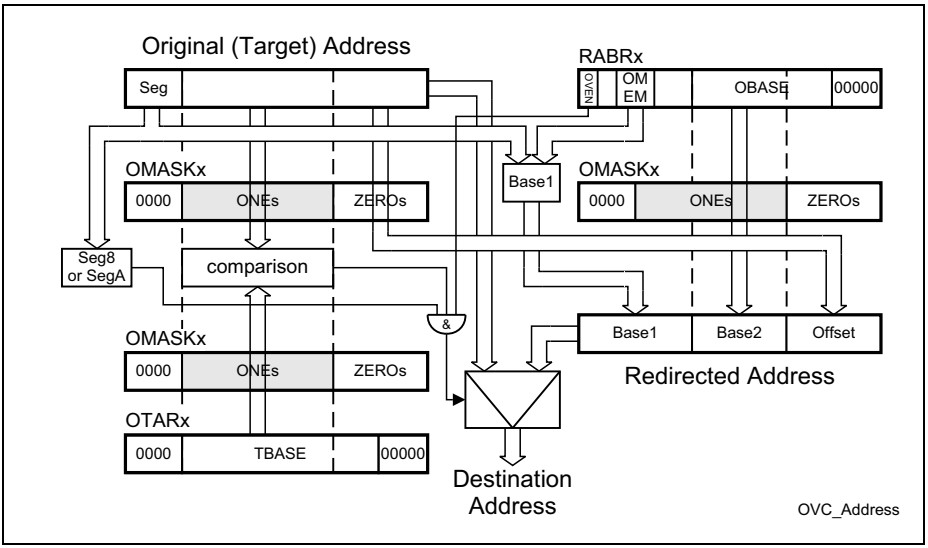


Figure 13-2 Address Redirection Process

Any data access to segment 8_H or segment A_H is checked against all the activated overlay blocks. For each activated overlay block, address bits 27..5 are compared with the target base address (OTARx), and this bit-wise comparison is qualified by the content of OMASKx register. Address bits participate in the comparison if the

corresponding OMASKx bits are set to one. The access is redirected, if all the address bits selected by OMASKx equal to the corresponding bits in OTARx register.

The address for redirection is constructed as follows:

- Address bits 31..22 are set according to the overlay memory selection (RABRx.OMEM) and the cache-ability of the original address.
- For address bits 21..5:
  - If the corresponding OMASKx bit is set, the address bit value is taken from RABRx.OBASE field;
  - If the corresponding OMASKx bit is cleared, the address bit value is taken from the original address.
- Address bits 4..0 are always taken directly from the original address.

If there is no redirection, the original address is used to perform the access.

Target address ranges for activated overlay blocks should not overlap or an exception may occur, see **Chapter13.6**.

13.2 Target Memories

Data access to any memory within the segment 8_H or segment A_H may be redirected to overlay memory. In particular, access to the following memories may be redirected:

- Program Flash;
- Data Flash;
- OLDA space;
- External EBU space.

13.2.1 Online Data Acquisition (OLDA) Space

Calibration is additionally supported by virtual OLDA memory range. The base address of the virtual OLDA memory range is A/8FE7 0000_H.

13.3 Overlay Memories

In the following, all the supported overlay memories are described. Note, that depending on the device type only a subset of the listed overlay memories is available. Overlay memory is selected independently for each block by RABRx.OMEM field value.

13.3.1 Local Memory

If present, the Local Memory (LMU) can be selected for overlay. The Local Memory is selected for overlay block x redirection, if RABRx.OMEM value is 6. The base address of the LMU is B/9000 0000_H. During address translation, the upper 10 address bits are set to $B0_H00_B$ (for target segment A_H) or to 90_H00_B (for target segment 8_H).

13.3.2 Emulation Memory

If present, the Emulation Memory (EMEM) can be selected for overlay. The Emulation Memory is selected for overlay block x redirection, if RABRx.OMEM value is 7. The base address of the Emulation Memory is B/9F00 0000_H. During address translation, the upper 10 address bits are set to BF_H00_B (for target segment A_H) or to $9F_H00_B$ (for target segment 8_H).

13.3.3 DSPR & PSPR Memory

Data Scratch Memory (DSPR) or Program Scratch Memory (PSPR) can be selected for overlay. The DSPR or PSPR is selected for overlay block x redirection, if RABRx.OMEM value is 0, 1, or 2. The base address is 7000 0000_H, 6000 0000_H, or 5000 0000_H. During address translation, the upper 10 address bits are set to 70_H00_B, 60_H00_B, or 50_H00_B. Depending on the value set in RABRx.OBASE either DSPR memory (starting at offset 0_H) or PSPR memory (starting at offset 10 0000_H) can be used.

13.4 Global Overlay Control

Overlay can be disabled or enabled individually for each core with OVCENABLE register. If OVCENABLE.OVENx bit is cleared no address redirection is permitted on Core x regardless of the remaining register settings. A write to OVCENABLE register does not change any of the remaining register values.

While each overlay block can be activated and deactivated individually by writing its RABRx.OVEN bit, a dedicated functionality is provided for concurrently activating and deactivating multiple blocks. This can be useful in maintaining data consistency across several memory regions. For the purpose of concurrent activation and deactivation overlay blocks are selected in two stages:

- The individual blocks for activation and deactivation are selected with OVCx_OSEL registers, for each core x independently;
- The set of cores is selected with OVCCON.CSEL field.

Multiple overlay blocks can be simultaneously activated or deactivated with OVCCON.OVSTRT bit. When OVCCON.OVSTRT bit is written with one:

- If OVCCON.CSELx bit is written with one, and OVCx_OSEL.SHOVENy bit value is one, overlay block y in core x is activated, and OVCx_RABRy.OVEN bit is set;
- If OVCCON.CSELx bit is written with one, and OVCx_OSEL.SHOVENy bit value is zero, overlay block y in core x is deactivated, and OVCx_RABRy.OVEN bit is cleared;
- If OVCCON.CSELx bit is written with zero, the overlay configuration in core x is not effected.

The actions listed above are executed concurrently. The overlay configuration is not changed otherwise. With this function it is possible to switch directly from one set of overlay blocks to another set of overlay blocks.

Multiple overlay blocks can be simultaneously deactivated with OVCCON.OVSTP bit. When OVCCON.OVSTP is written with one:

- If OVCCON.CSELx bit is written with one, all the overlay blocks in core x are deactivated, and all OVCx_RABRy.OVEN bits are cleared;
- If OVCCON.CSELx bit is written with zero, the overlay configuration in core x is not effected.

The actions listed above are executed concurrently. The overlay configuration is not changed otherwise.

Note: Overlay should not be enabled or disabled using global OVCENABLE register if any of the blocks are enabled with RABRx.OVEN. Instead, OVSTRT or OVSTP should be used if concurrent block enabling or disabling is required.

When OVCCON.DCINVAL is written with one, all the unmodified (clean) data cache lines in the selected cores are invalidated. The data cache lines containing modified (dirty) data are not effected. The cores not selected with OVCCON.CSEL field are not effected. Data Cache invalidation can be combined with OVSTRT or OVSTP action. This function helps to assure that the CPU can access the new data after the overlay blocks have been activated or deactivated.

Note: OVCCON.CSEL field is written together with OVSTRT, OVSTP and DCINVAL bits in the same register, and only impacts any action triggered by the same write. CSEL, OVSTRT, OVSTP and DCINVAL do not retain the written value and always read as zero.

The OVCCON.OVCONF user control flag is provided, together with its protection bit OVCCON.POVCONF. These bits do not impact the overlay functionality.

When OVCCON register is written with CSELx set and either OVSTRT or OVSTP set, and at the same time OVCx_RABRy register is written, the resulting value of OVCx_RABRy.OVEN bit is not defined. Possibility of such simultaneous access should be avoided.

OVSTRT, OVSTP and DCINVAL actions are not performed when TriCore is in IDLE state.

13.4.1 Global Overlay Control Synchronisation

When OVSTRT, OVSTP or DCINVAL action is requested its execution may be delayed to prevent changing Overlay configuration during an ongoing data load.

Sufficient time should be allowed after an action request, before another action is requested. If a new action is requested while previous action is still pending (due to synchronisation with CPU loads) some actions may be lost.

13.5 Overlay Configuration Change

Overlay block should be disabled, by clearing its RABRx.OVEN bit, before any changes are made to OTARx, OMASKx or RABRx registers. Otherwise, unintended access redirections may occur. Overlay block should only be enabled, if the target address, the overlay memory selection, the redirection address and the mask have all been configured with intended values.

Note: The Overlay Control does not prevent configuring the translation logic incorrectly. In particular, redirection to not implemented or forbidden address range is not prevented.

Special care needs to be taken to synchronise Overlay redirection change to the executed instruction stream if data consistency is required.

External accesses may be buffered in the CPU. External accesses that have not been completed may still be affected by overlay configuration changes. Therefore, it is advised to ensure completion of all pending accesses (for example, by executing DSYNC instruction) before any overlay range is acitvated of deactivated.

When overlay block is enabled and the same memory location is written through the target address space and read through the redirected address space, or vice-versa, the access synchronisation need to be enforced (with DSYNC and data cache writeback if applicable).

13.6 Access Protection, Attributes, Concurrent Matches

When data access is redirected by Overlay, access protection is applied as follows:

- Target address is subject to CPU Memory Protection (MPU) validation;
- Redirected address is subject to Safety Protection validation.

Physical Memory Attributes for the redirected access are determined basing on the Target Address segment (see CPU Physical Memory Attributes chapter).

Concurrent matches in more than one enabled overlay block are not supported. When an address matches two, or more, of the enabled overlay blocks, an exception is raised and the memory access is not performed. A load operation with multiple matches on overlay ranges, raises a Data Access Synchronous Error (DSE) trap, and a store operation raises Data Access Asynchronous Error (DAE) trap. In such case, relevant trap information registers: Data Synchronous Trap Register (DSTR), Data Asynchronous Trap Register (DATR), and Data Error Address Register (DEADD) are updated, see DMI Registers chapter for more information.

13.7 Overlay Control Registers

OVC block control registers are located in each module that supports data access overlay. OVC global control registers are located in SCU. OVC register access can be restricted by Safety Register Protection.

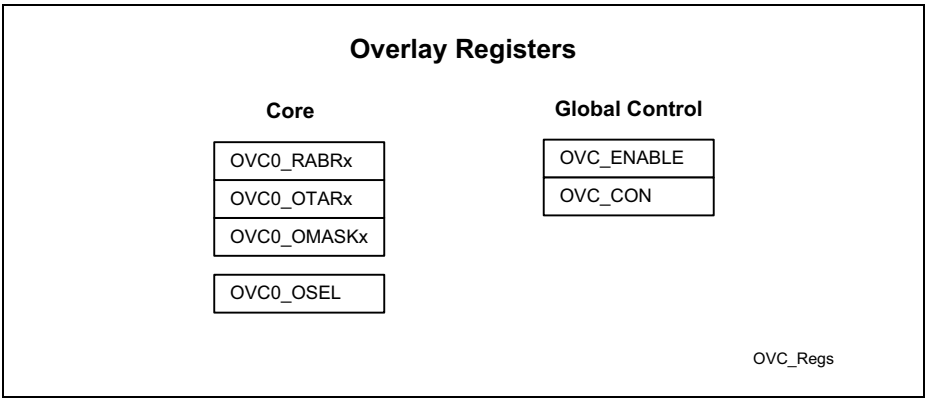


Figure 13-3 Overlay Register Overview

Registers OVCENABLE and OVCCON are described in SCU Chapter.

Table 13-1 Registers Address Space of OVC Registers

Module	Base Address	End Address	Note
OVC0	F880 FB00$_H$	F880 FCFF$_H$	Core0 Overlay

Table 13-2 Registers Overview, Overlay Block Control

Register[1] Short Name	Register Long Name	Offset Address	Access Mode[2]		Descript-ion see
			Read	Write	
OSEL	Overlay Range Select Register	0000_H	U, SV, 32	P, SV, 32	Page 13-10
RABRx	Redirected Address Base Register x (x = 0-7)	$0010_H + x * C_H$	U, SV, 32	P, SV, 32	Page 13-11
OTARx	Overlay Target Address Register x (x = 0-7)	$0014_H + x * C_H$	U, SV, 32	P, SV, 32	Page 13-13
OMASKx	Overlay Mask Register x (x = 0-7)	$0018_H + x * C_H$	U, SV, 32	P, SV, 32	Page 13-14

1) The OVC register short names are extended with the module name prefix “OVCi_”, where i is core number.

2) Symbol P: Write access protected with Safety Protection
Symbol SV: Access permitted in Supervisor Mode
Symbol U: Access permitted in User Mode 0 or 1
Symbol 32: Only 32-bit word access is permitted

13.7.1 Block control registers

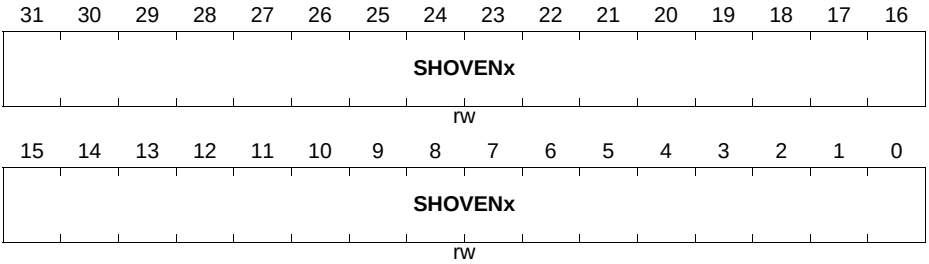
For each of the 8 overlay memory blocks (indicated by index x), three registers control the overlay operation and the memory selection:

- Redirected Address Base Register RABRx, which selects the overlay memory, holds the block base address within this memory, and contains block enable bit.
- Overlay Target Address Register OTARx, which holds the base address of the memory block being overlaid.
- Overlay Mask Register OMASKx, which determines which bits (from RABRx) are used for the base address (of overlay memory and block) and which bits (of original data address) are directly used as offset within the block.

Additionally, Overlay Range Select Registers OSEL determines which blocks are to be enabled and which blocks are to be disabled when OVCCON.OVSTRT bit is set.

All overlay block control registers are reset to their default values with the application reset. A special debug reset is not considered.

OSEL
Overlay Range Select Register (0000_H) Reset Value: $0000\ 0000_H$

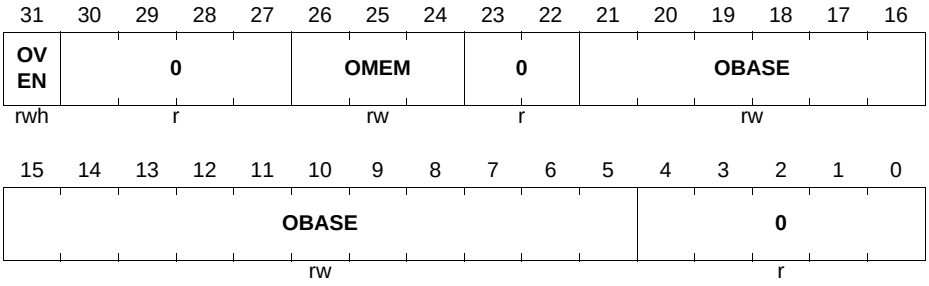


Field	Bits	Type	Description
SHOVENx (x=0-7)	x	rw	**Shadow Overlay Enable x** 0_B Overlay block x is disabled when OVCCON.OVSTRT is set. 1_B Overlay block x is enabled when OVCCON.OVSTRT is set. One enable bit is provided for each of the 8 overlay blocks (indicated by index x).

*Note: See **Chapter13.4.1** for more information on using OSEL register.*

RABRx (x=0-7)
Redirected Address Base Register x

(10_H+x*C_H) **Reset Value: 0000 0000_H**

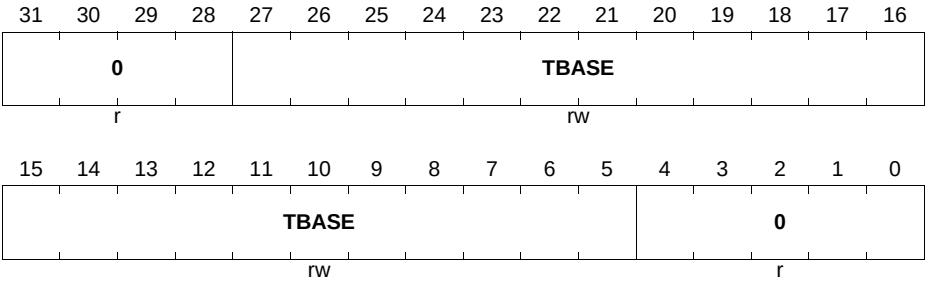


Field	Bits	Type	Description
OVEN	31	rwh	**Overlay Enabled** This bit controls whether the overlay function of overlay block x is enabled. 0_B Overlay function of block x is disabled. 1_B Overlay function of block x is enabled. This bit can also be changed when OVCCON.OVSTP or OVCCON.OVSTRT is set. See OVCCON register description.
OMEM	[26:24]	rw	**Overlay Memory Select** Selects overlay memory used for redirection. 0_H Redirection to Core 0 DSPR/PSPR memory $1..5_H$ Reserved, do not use 6_H Redirection to LMU 7_H Redirection to EMEM
OBASE	[21:5]	rw	**Overlay Block Base Address** Bits 21..5 of the base address the overlay memory block in the overlay memory. If the corresponding bit in OMASK register is set to one, OBASE bit value is used in the redirection address. If the corresponding bit in OMASK register is set to zero, OBASE bit value is ignored.

Field	Bits	Type	Description
0	[30:27] [23:22] [4:0]	r	**Fixed Value** Read as 0; should be written with 0.

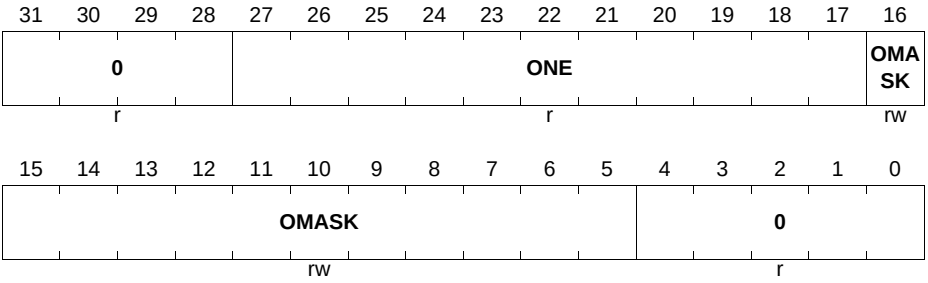
OTARx (x=0-7)
Overlay Target Address Register x

(14_H+x*C_H) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
TBASE	[27:5]	rw	**Target Base** This field holds the base address of the overlay memory block in the target memory. If the corresponding bit in OMASK register is set to one TBASE bit value is used in the address match. If the corresponding bit in OMASK register is set to zero TBASE bit value is ignored.
0	[31:28] [4:0]	r	**Reserved** Reads as 0; should be written with 0.

OMASKx (x=0-7)
Overlay Mask Register x **(18_H+x*C_H)** **Reset Value: 0FFF FFE0$_H$**



Field	Bits	Type	Description
OMASK	[16:5]	rw	**Overlay Address Mask** This bitfield determines the overlay block size and the bits used for address comparison and translation. 000000000000_B, 128 Kbyte block size 100000000000_B, 64 Kbyte block size 110000000000_B, 32 Kbyte block size [...] 111111111110_B, 64 byte block size 111111111111_B, 32 byte block size “Zero” bits determine the corresponding address bits which are not used in the address comparison and thus determine the block size; corresponding final address bits are derived from the original data address. “One” bits determine the corresponding address bits which are used for the address comparison; corresponding final address bits are derived from RABRx register in case of address match.
ONE	[27:17]	r	**Fixed “1” Values** Corresponding address bits are participating in the address comparison. Corresponding final address bits are taken from RABRx.

Field	Bits	Type	Description
0	[4:0] [31:28]	r	**Fixed “0” Values** Corresponding address bits are not used in the address comparison. Corresponding final address bits are taken from the original data address.

13.8 Global overlay control registers

Two registers globally control the overlay operation for all the cores:

- Overlay Enable Register OVCENABLE can be used to disable or enable data access overlay individually for each core;
- Overlay Control Register OVCCON can be used to perform the following action on selected set of cores:
  – concurrently enable / disable selected overlay blocks,
  – concurrently disable overlay blocks,
  – invalidate data cache.

.

14 General Purpose I/O Ports and Peripheral I/O Lines (Ports)

The TC21x/TC22x/TC23x has digital General Purpose Input/Output (GPIO) port lines which are connected to the on-chip peripheral units.

Attention: Please refer to the pin definition and functions chapters for port input and output signal mapping.

14.1 Basic Port Operation

Figure 14-1 is a general block diagram of a TC21x/TC22x/TC23x GPIO port slice.

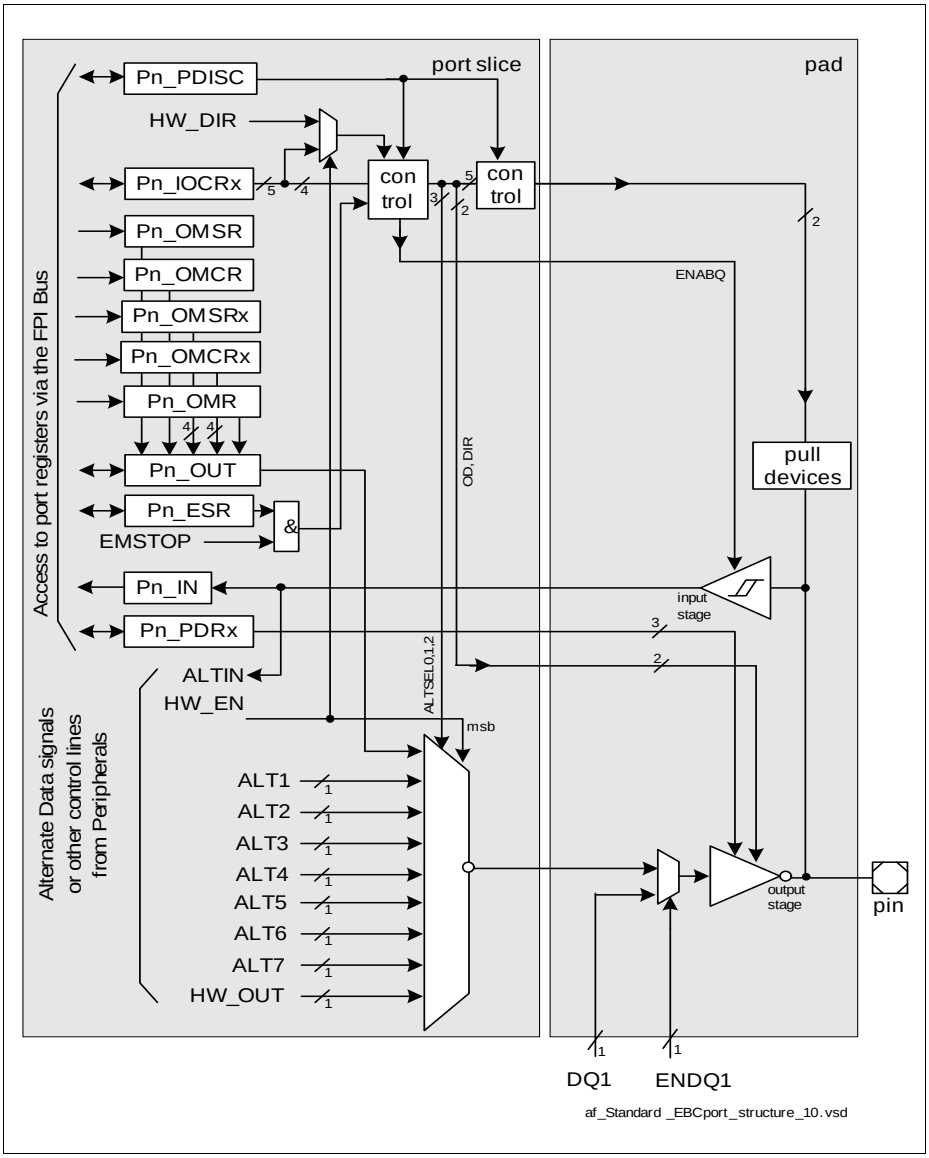


Figure 14-1 General Structure of a Port Pin

Each port line has a number of control and data bits, enabling very flexible usage of the line. Each port pin can be configured for input or output operation. In input mode (default after reset), the output driver is switched off (high-impedance). The actual voltage level present at the port pin is translated into a logical 0 or 1 via a Schmitt-Trigger device and can be read via the read-only register Pn_IN. Input signals are connected directly to the various inputs of the peripheral units (AltDataIn). The function of the input line from the pin to the input register Pn_IN and to AltDataIn is independent of whether the port pin operates as input or output. This means that when the port is in output mode, the level of the pin can be read by software via Pn_IN or a peripheral can use the pin level as an input.

In output mode, the output driver is activated and drives the value supplied through the multiplexer to the port pin. Switching between input and output mode is accomplished through the Pn_IOCR register, which enables or disables the output driver. If a peripheral unit uses a GPIO port line as a bi-directional I/O line, register Pn_IOCR has to be written for input or output selection. The Pn_IOCR register further controls the driver type of the output driver, and determines whether an internal weak pull-up, pull-down, or without input pull device is alternatively connected to the pin when used as an input. This offers additional advantages in an application.

The output multiplexer in front of the output driver selects the signal source for the GPIO line when used as output. If the pin is used as general-purpose output, the multiplexer is switched by software (Pn_IOCR register) to the Output Data Register Pn_OUT. Software can set or clear the bit in Pn_OUT through separate Pn_OMSR or Pn_OMCR registers. The set or clear operations for the bits in Pn_OUT can also be done for up to four bits per register in Pn_OMSRx and Pn_OMCRx (x=0,4,8,12). Alternatively, the set, clear or toggle function can be achieved through Pn_OMR, where adjacent pins within the same port can be set, cleared or toggled within one write operation. The manipulation of the control bits in these registers can directly influence the state of the port pin. If the on-chip peripheral units use the pin for output signals, the alternate output lines ALT1 to ALT7 can be switched via the multiplexer to the output driver. The data written into the output register Pn_OUT by software can be used as input data to an on-chip peripheral. This enables, for example, peripheral tests via software without external circuitry.

When selected as general-purpose output line, the logic state of each port pin can be changed individually by programming the pin-related bits in the Output Modification Set Register Pn_OMSR, Output Modification Set Register x Pn_OMSRx (x=0,4,8,12), Output Modification Clear Register Pn_OMCR, Output Modification Clear Register x Pn_OMCRx (x=0,4,8,12) or Output Modification Register, OMR. The bits in Pn_OMSR/Pn_OMSRx and Pn_OMCR/Pn_OMCRx make it possible to set and clear the bits in the Pn_OUT register. While the bits in Pn_OMR allows the bits in Pn_OUT to be set, cleared, toggled or remain unchanged.

When selected as general-purpose output line, the actual logic level at the pin can be examined through reading Pn_IN and compared against the applied output level (either applied through software via the output register Pn_OUT, or via an alternate output

function of a peripheral unit). This can be used to detect some electrical failures at the pin caused through external circuitry. In addition, software-supported arbitration schemes can be implemented in this way using the open-drain configuration and an external wired-And circuitry. Collisions on the external communication lines can be detected when a high level (1) is output, but a low level (0) is seen when reading the pin value via the input register Pn_IN.

All digital GPIO lines of the TC21x/TC22x/TC23x has an emergency stop logic. This logic makes it possible to individually disconnect outputs and put them onto a well defined logic state in an emergency case. In an emergency case, the pin is switched to input function in tri-state mode. The Emergency Stop Register Pn_ESR determines whether an output is enabled or disabled in an emergency case.

14.2 Description Scheme for the Port IO Functions

The following two general building block can be used to describe each GPIO pin:

Table 14-1 Port x Input/Output Functions

Port Pin	I/O	Select	Connected Signal(s)	From / to Module
Px.y	Input		Signal(s)	module(s)
	Output	GPIO	Signal	module
		ALT1	Signal	module
		ALT2	Signal	module
		ALT3	Signal	module
		ALT4	Signal	module
		ALT5	Signal	module
		ALT6	Signal	module
		ALT7	Signal	module
	HW_DIR	HW_Out	Signal	module; group En

or:

Table 14-2 Port x Functions

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
Px.y	I	GPIO	Px_IN.Py	Px_IOCRz. PC0	$0XXX0_B$
	O	GPIO	Px_OUT.Py		$1X000_B$
					$1X001_B$
					$1X010_B$
					$1X011_B$
					$1X100_B$
					$1X101_B$
					$1X110_B$
					$1X111_B$
	HW_DIR	module; group En	Signal		HW_DIR

- HW_DIR:
  The type Alternate Direction signal which is needed if HW_En is active:
  – Out -always output
    DIRx - the pins in one port having the same DIRx (x=0, 1, 2, ...), are controlled as a group by a dedicated HW_DIR signal.
    SDIR- Single DIR- the pin is controlled by its own, dedicated, single HW_DIR signal.
- grouping indicates if the respective pin is controlled by hardware:
  – ENx - the pins in one port having the same ENx (x=0, 1, 2, ...), are controlled as a group by a dedicated HW_EN signal.
  – SEN - Single EN - the pin is controlled by its own, dedicated, single HW_EN signal
- Digital port slices with HW_DIR defined are the ports described in **Figure 14-1**.

Note: Emergency Stop has higher priority than the HW_EN signal. Emergency Stop is functional when the pins are set in the GPIO mode.

Note: HW_DIR signal, output case, switches the pad to push-pull output state.
HW_DIR signal, input case, switches the pad to the input state with pull-up/down setting as defined by the IOCR register.

14.3 Port Register Description

The individual control and data bits of each GPIO port are implemented in a number of registers. The registers are used to configure and use the port as general-purpose I/O or alternate function input/output. For some ports, not all registers are implemented. The availability of the registers in the specific ports is described separately.

Note: Parallel requests from on chip bus masters to the ports module are executed sequentially via the on chip bus system. Read-modify-write feature provides an atomic read/write sequence where no other master can access the ports module in between the operations.

Port Register Overview

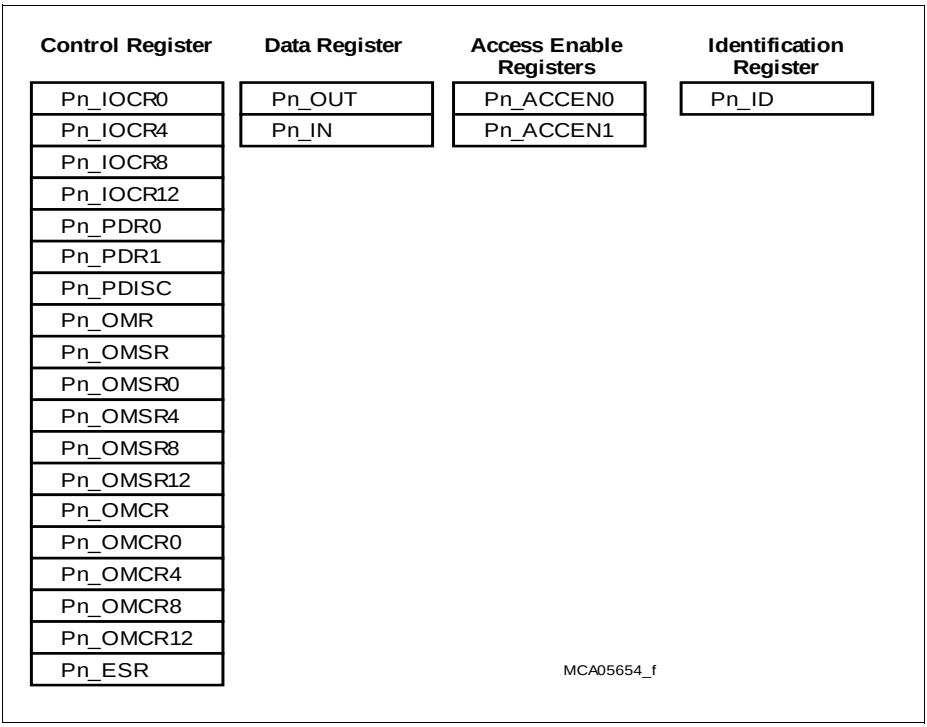


Figure 14-2 Port Registers

Table 14-3 Registers Address Space

Module	Base Address	End Address	Note
P00	F003 A000$_H$	F003 A0FF$_H$	13 pins; pins[12:0]
P02	F003 A200$_H$	F003 A2FF$_H$	9 pins; pins[8:0]
P10	F003 B000$_H$	F003 B0FF$_H$	5 pins; pins[6:5], [3:1]
P11	F003 B100$_H$	F003 B1FF$_H$	8 pins; pins [3:2], 6, [12:8]
P13	F003 B300$_H$	F003 B3FF$_H$	4 pins; pins[3:0]
P14	F003 B400$_H$	F003 B4FF$_H$	9 pins; pins[8:0]
P15	F003 B500$_H$	F003 B5FF$_H$	9 pins; pins[8:0]
P20	F003 C000$_H$	F003 C0FF$_H$	12 pins; pins [14:6], [3:2], 0
P21	F003 C100$_H$	F003 C1FF$_H$	6 pins; pins[7:2]
P22	F003 C200$_H$	F003 C2FF$_H$	5 pins; pins[4:0]
P23	F003 C300$_H$	F003 C3FF$_H$	1 pin; pin1
P33	F003 D300$_H$	F003 D3FF$_H$	13 pins; pins[12:0]
P34	F003 D400$_H$	F003 D4FF$_H$	4 pins; pins[3:0]
P40	F003 E000$_H$	F003 E0FF$_H$	12 pins; pins[11:0]
P41	F003 E100$_H$	F003 E1FF$_H$	12 pins; pins[11:0]

Table 14-4 Registers Overview

Register Short Name	Register Long Name	Offset Address	Access Mode		Reset	Desc. see
			Read	Write		
Pn_OUT	Port n Output Register	0000$_H$	U, SV	U, SV, P	Application Reset	Page 14-24
Pn_OMR	Port n Output Modification Register	0004$_H$	U, SV	U, SV, P	Application Reset	Page 14-26
ID	Module Identification Register	0008$_H$	U, SV	BE	Application Reset	Page 14-12

Table 14-4 Registers Overview (cont'd)

Register Short Name	Register Long Name	Offset Address	Access Mode		Reset	Desc. see
			Read	Write		
Pn_IOCR0	Port n Input/Output Control Register 0	0010_H	U, SV	U, SV, P	Application Reset	Page 14-13
Pn_IOCR4	Port n Input/Output Control Register 4	0014_H	U, SV	U, SV, P	Application Reset	Page 14-13
Pn_IOCR8	Port n Input/Output Control Register 8	0018_H	U, SV	U, SV, P	Application Reset	Page 14-13
Pn_IOCR12	Port n Input/Output Control Register 12	$001C_H$	U, SV	U, SV, P	Application Reset	Page 14-13
–	Reserved	0020_H	BE	BE	–	–
Pn_IN	Port n Input Register	0024_H	U, SV	BE	Application Reset	Page 14-39
–	Reserved	0028_H-$003C_H$	BE	BE	–	–
Pn_PDR0	Port n Pad Driver Mode 0 Register	0040_H	U, SV	SV, E, P	Application Reset	Page 14-19
Pn_PDR1	Port n Pad Driver Mode 1 Register	0044_H	U, SV	SV, E, P	Application Reset	Page 14-19
–	Reserved	0048_H-$004C_H$	BE	BE	–	–
Pn_ESR	Port n Emergency Stop Register	0050_H	U, SV	SV, E, P	Application Reset	Page 14-38
–	Reserved	0054_H-$005C_H$	–	–	–	–
Pn_PDISC	Port n Pin Function Decision Control Register	0060_H	U, SV	SV, E, P	Application Reset	Page 14-22
Pn_PCSR	Port n Pin Controller Select Register	0064_H	U, SV	SV, SE	Application Reset	Page 14-23
–	Reserved	0064_H	–	–	–	–
–	Reserved	0068_H-$006C_H$	BE	BE	–	–

Table 14-4 Registers Overview (cont'd)

Register Short Name	Register Long Name	Offset Address	Access Mode		Reset	Desc. see
			Read	Write		
Pn_OMSR0	Port n Output Modification Set Register 0	0070_H	U, SV	U, SV	Application Reset	Page 14-29
Pn_OMSR4	Port n Output Modification Set Register 4	0074_H	U, SV	U, SV	Application Reset	Page 14-29
Pn_OMSR8	Port n Output Modification Set Register 8	0078_H	U, SV	U, SV	Application Reset	Page 14-29
Pn_OMSR12	Port n Output Modification Set Register 12	$007C_H$	U, SV	U, SV	Application Reset	Page 14-29
Pn_OMCR0	Port n Output Modification Clear Register 0	0080_H	U, SV	U, SV	Application Reset	Page 14-34
Pn_OMCR4	Port n Output Modification Clear Register 4	0084_H	U, SV	U, SV	Application Reset	Page 14-34
Pn_OMCR8	Port n Output Modification Clear Register 8	0088_H	U, SV	U, SV	Application Reset	Page 14-34
Pn_OMCR12	Port n Output Modification Clear Register 12	$008C_H$	U, SV	U, SV	Application Reset	Page 14-34
Pn_OMSR	Port n Output Modification Set Register	0090_H	U, SV	U, SV	Application Reset	Page 14-28
Pn_OMCR	Port n Output Modification Clear Register	0094_H	U, SV	U, SV	Application Reset	Page 14-33
–	Reserved	0098_H-$009C_H$	BE	BE	–	–

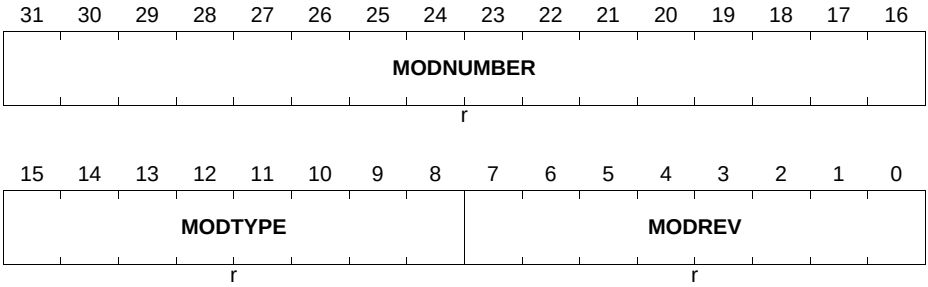
Table 14-4 Registers Overview (cont'd)

Register Short Name	Register Long Name	Offset Address	Access Mode		Reset	Desc. see
			Read	Write		
–	Reserved	$00A0_H$-$00BC_H$	U, SV	SV, E, P	–	–
–	Reserved	$00C0_H$-$00F4_H$	BE	BE	–	–
Pn_ ACCEN1	Port n Access Enable Register 1	$00F8_H$	U, SV	SV, SE	Application Reset	Page 14-43
Pn_ ACCEN0	Port n Access Enable Register 0	$00FC_H$	U, SV	SV, SE	Application Reset	Page 14-41

14.3.1 Module Identification Register

The module Identification Register ID contains read-only information about the module version.

ID
Identification Register **(08_H)** **Reset Value: $00C8\ C0XX_H$**



Field	Bits	Type	Description
MODREV	[7:0]	r	**Module Revision Number** This bit field indicates the revision number of the TC21x/TC22x/TC23x module (01_H = first revision).
MODTYPE	[15:8]	r	**Module Type** This bit field is $C0_H$. It defines a 32-bit module
MODNUMBER	[31:16]	r	**Module Number** This bit field defines the module identification number. The value for the Ports module is $00C8_H$

14.3.2 Port Input/Output Control Registers

The port input/output control registers select the digital output and input driver functionality and characteristics of a GPIO port pin. Port direction (input or output), pull-up, pull-down, or no pull devices for inputs, and push-pull or open-drain functionality for outputs can be selected by the corresponding bit fields PCx (x = 0-15). Each 32-bit wide port input/output control register controls four GPIO port lines:

Register Pn_IOCR0 controls the Pn.[3:0] port lines
Register Pn_IOCR4 controls the Pn.[7:4] port lines
Register Pn_IOCR8 controls the Pn.[11:8] port lines
Register Pn_IOCR12 controls the Pn.[15:12] port lines

The diagrams below show the register layouts of the port input/output control registers with the PCx bit fields. One PCx bit field controls exactly one port line Pn.x.

P00_IOCR0
Port 00 Input/Output Control Register 0
(10_H) Reset Value: 0000 0000$_H$

P02_IOCR0
Port 02 Input/Output Control Register 0
(10_H) Reset Value: 0000 0000$_H$

P13_IOCR0
Port 13 Input/Output Control Register 0
(10_H) Reset Value: 0000 0000$_H$

P14_IOCR0
Port 14 Input/Output Control Register 0
(10_H) Reset Value: 1010 0000$_H$

P15_IOCR0
Port 15 Input/Output Control Register 0
(10_H) Reset Value: 0000 0000$_H$

P22_IOCR0
Port 22 Input/Output Control Register 0
(10_H) Reset Value: 0000 0000$_H$

Pn_IOCR0 (n=33-34)
Port n Input/Output Control Register 0
($F003\ B210_H + n*100_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PC3					0			PC2					0		
rw					r			rw					r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC1					0			PC0					0		
rw					r			rw					r		

Field	Bits	Type	Description
PC0, **PC1,** **PC2,** **PC3**	[7:3], [15:11], [23:19], [31:27]	rw	**Port Control for Port n Pin 0 to 3** This bit field determines the Port n line x functionality (x = 0-3) according to the coding table (see **Table 14-5**).

Field	Bits	Type	Description
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

P00_IOCR4
Port 00 Input/Output Control Register 4
(14_H) **Reset Value:0000 0000$_H$**

P02_IOCR4
Port 02 Input/Output Control Register 4
(14_H) **Reset Value: 0000 0000$_H$**

P14_IOCR4
Port 14 Input/Output Control Register 4
(14_H) **Reset Value: 0010 0000$_H$**

P15_IOCR4
Port 15 Input/Output Control Register 4
(14_H) **Reset Value: 0000 0000$_H$**

P21_IOCR4
Port 21 Input/Output Control Register 4
(14_H) **Reset Value: 0010 0000$_H$**

P33_IOCR4
Port 33 Input/Output Control Register 4
(14_H) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PC7					**0**			**PC6**					**0**		
rw					r			rw					r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC5					**0**			**PC4**					**0**		
rw					r			rw					r		

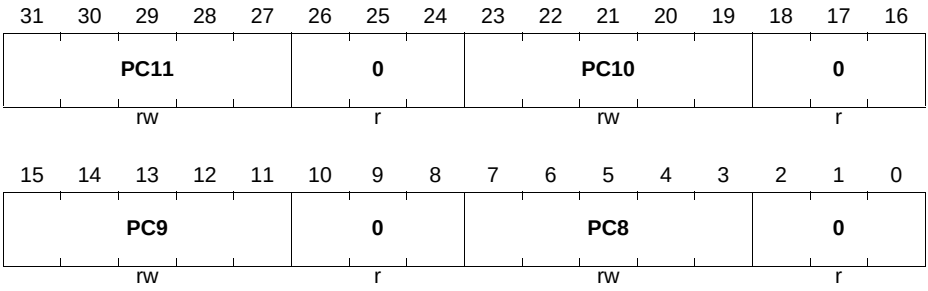
Field	Bits	Type	Description
PC4, PC5, PC6, PC7	[7:3], [15:11], [23:19], [31:27]	rw	**Port Control for Port n Pin 4 to 7** This bit field determines the Port n line x functionality (x = 4-7) according to the coding table (see **Table 14-5**).
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

P00_IOCR8
Port 00 Input/Output Control Register 8
(18_H) **Reset Value: 0000 0000$_H$**

P11_IOCR8
Port 11 Input/Output Control Register 8
(18_H) **Reset Value: 0000 0000$_H$**

P20_IOCR8
Port 20 Input/Output Control Register 8
(18_H) **Reset Value: 0000 0000$_H$**

P33_IOCR8
Port 33 Input/Output Control Register 8
(18_H) **Reset Value: 0000 0000$_H$**

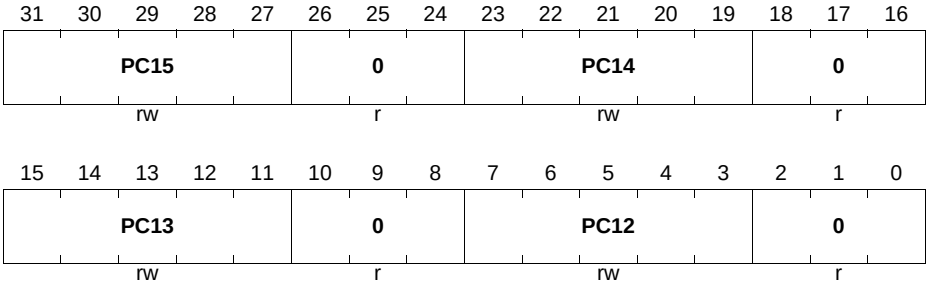


Field	Bits	Type	Description
PC8, PC9, PC10, PC11	[7:3], [15:11], [23:19], [31:27]	rw	**Port Control for Port n Pin 8 to 11** This bit field determines the Port n line x functionality (x = 8-11) according to the coding table (see **Table 14-5**).

Field	Bits	Type	Description
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

Pn_IOCR12
Port n Input/Output Control Register 12

($1C_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
PC12, PC13, PC14, PC15	[7:3], [15:11], [23:19], [31:27]	rw	**Port Control for Port n Pin 12 to 15** This bit field determines the Port n line x functionality (x = 12-15) according to the coding table (see **Table 14-5**).
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

The structure with one control bit field for each port pin located in different register bytes offers the possibility to configure the port pin functionality of a single pin with byte-oriented accesses without accessing the other PCx bit fields.

Port Control Coding

Table 14-5 describes the coding of the PCx bit fields that determine the port line functionality.

Table 14-5 PCx Coding

PCx[4:0]	I/O	Characteristics	Selected Pull-up / Pull-down / Selected Output Function
$0XX00_B$	Input[1)]	–	No input pull device connected, tri-state mode
$0XX01_B$			Input pull-down device connected
$0XX10_B$			Input pull-up device connected
$0XX11_B$			No input pull device connected, tri-state mode
10000_B	Output	Push-pull	General-purpose output
10001_B			Alternate output function 1
10010_B			Alternate output function 2
10011_B			Alternate output function 3
10100_B			Alternate output function 4
10101_B			Alternate output function 5
10110_B			Alternate output function 6
10111_B			Alternate output function 7
11000_B		Open-drain	General-purpose output
11001_B			Alternate output function 1
11010_B			Alternate output function 2
11011_B			Alternate output function 3
11100_B			Alternate output function 4
11101_B			Alternate output function 5
11110_B			Alternate output function 6
11111_B			Alternate output function 7

1) Only input functions apply for P40 and P41.

14.3.3 Pad Driver Mode Register

Overview

The pad structure of the TC21x/TC22x/TC23x GPIO lines offers the possibility to select the output driver strength, the slew rate / edge control. These two parameters are controlled by the PDx bit fields in the pad driver mode registers Pn_PDR0/1 for output modes..

The assignment of each port pin to one of these pad classes is shown in the port configuration figures. Further details about pad driver classes that are available in the TC21x/TC22x/TC23x are summarized in the Target Data Sheet/Data Sheet (TBD).

Depending on the assigned pad class, the 3-bit wide pad driver mode selection bit fields PDx in the pad driver mode registers Pn_PDRx make it possible to select the port line functionality as shown in **Table 14-6**.

Table 14-6 Pad Driver Mode Selection

PDx.2	PDx.1	PDx.0	Functionality
X	0	0	Speed grade 1
X	0	1	Speed grade 2
X	1	0	Speed grade 3
X	1	1	Speed grade 4

Note: TC21x/TC22x/TC23x Data Sheet describes the DC characteristics of all pad classes.

Pad Driver Mode Registers

This is the general description of the PDR registers. Each port contains its own specific PDR registers, described additionally at each port, that can contain between one and eight PDx fields for PDR0 and PDR1 registers, respectively. Each PDx field controls 1 pin. For coding of PDx, see **Table 14-6**.

The reset value of Pn_PDR0/1registers is 3333 3333_H.

P00_PDR0
Port 00 Pad Driver Mode 0 Register (40_H) Reset Value: 3333 3333$_H$
P02_PDR0
Port 02 Pad Driver Mode 0 Register (40_H) Reset Value: 3333 3333$_H$
Pn_PDR0 (n=14-15)
Port n Pad Driver Mode 0 Register($F003\ A640_H+n*100_H$) Reset Value: 3333 3333$_H$
P33_PDR0
Port 33 Pad Driver Mode 0 Register (40_H) Reset Value: 3333 3333$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PL7		PD7		PL6		PD6		PL5		PD5		PL4		PD4	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PL3		PD3		PL2		PD2		PL1		PD1		PL0		PD0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
PD0, **PD1,** **PD2,** **PD3,** **PD4,** **PD5,** **PD6,** **PD7**	[2:0], [6:4], [10:8], [14:12], [18:16], [22:20], [26:24], [30:28]	rw	**Pad Driver Mode for Port n Pin 0 to 7** See **Table 14-6**.
PL0, **PL1,** **PL2,** **PL3,** **PL4,** **PL5,** **PL6,** **PL7**	3, 7, 11, 15, 19, 23, 27, 31	rw	**Reserved** Read as 0, returns values last written, must be written with 0.

Pn_PDR1

Port n Pad Driver Mode 1 Register (44_H) Reset Value: 3333 3333_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PL15	**PD15**			**PL14**	**PD14**			**PL13**	**PD13**			**PL12**	**PD12**		
rw	rw			rw	rw			rw	rw			rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PL11	**PD11**			**PL10**	**PD10**			**PL9**	**PD9**			**PL8**	**PD8**		
rw	rw			rw	rw			rw	rw			rw	rw		

Field	Bits	Type	Description
PD8, **PD9,** **PD10,** **PD11,** **PD12,** **PD13,** **PD14,** **PD15**	[2:0], [6:4], [10:8], [14:12], [18:16], [22:20], [26:24], [30:28]	rw	**Pad Driver Mode for Port n Pin 8 to 15** See **Table 14-6**.
PL8, **PL9,** **PL10,** **PL11,** **PL12,** **PL13,** **PL14,** **PL15**	3, 7, 11, 15, 19, 23, 27, 31	rw	**Reserved** Read as 0, returns value last written, must be written with 0.

14.3.4 Pin Function Decision Control Register

Pin Function Decision Control Register

The pad structure of the TC21x/TC22x/TC23x GPIO lines offers the possibility to select digital input or analog ADC input functionalities. For the selection of digital input, the parameters defined for Class S pads must be met. This feature can be controlled by individual bits in the Pn_PDISC register, independently from input/output and pull-up/pull-down control functionality as programmed in the Pn_IOCR register. One Pn_PDISC register is assigned to each port.

P40_PDISC
Port 40 Pin Function Decision Control Register(60_H) **Reset Value: 0000 0FFF$_H$**
P41_PDISC
Port 41 Pin Function Decision Control Register(60_H) **Reset Value: 0000 0FFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDIS 15	**PDIS 14**	**PDIS 13**	**PDIS 12**	**PDIS 11**	**PDIS 10**	**PDIS 9**	**PDIS 8**	**PDIS 7**	**PDIS 6**	**PDIS 5**	**PDIS 4**	**PDIS 3**	**PDIS 2**	**PDIS 1**	**PDIS 0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PDISx (x = 0-15)	x	rw	**Pin Function Decision Control for Pin x** This bit selects the function of the port pad. 0_B Pad Pn.x is selected for digital input x. 1_B Pad Pn.x is selected for ADC analog input x.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

14.3.5 Pin Controller Select Register

PCSR register is used to enable or disable the VADC Multiplexer Diagnostics (MD) feature.

P40_PCSR
Port 40 Pin Controller Select Register (64_H) **Reset Value: 0000 0000$_H$**
P41_PCSR
Port 41 Pin Controller Select Register (64_H) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LCK								**0**							
rh								r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEL 15	**SEL 14**	**SEL 13**	**SEL 12**	**SEL 11**	**SEL 10**	**SEL 9**	**SEL 8**	**SEL 7**	**SEL 6**	**SEL 5**	**SEL 4**	**SEL 3**	**SEL 2**	**SEL 1**	**SEL 0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
SELx (x = 0-15)	x	rw	**Pin Controller Select for Pin x** This bit enables or disables the VADC Multiplexer Diagnostics (MD) feature. 0_B Disable VADC MD feature of pin x. 1_B Enable VADC MD feature of pin x.
LCK	31	rh	**Lock Status** This bit indicates if the register can be updated with a new value or if the register is locked and a write action from the bus has no effect. In this device without function (LCK stays 0_B). 0_B The register is unlocked and can be updated. 1_B The register is locked and can not be updated.
0	[30:16]	r	**Reserved** Read as 0; should be written with 0.

14.3.6 Port Output Register

The port output register determines the value of a GPIO pin when it is selected by Pn_IOCRx as output. Writing a 0 to a Pn_OUT.Px (x = 0-15) bit position delivers a low level at the corresponding output pin. A high level is output when the corresponding bit is written with a 1. Note that the bits of Pn_OUT.Px can be individually set or cleared by writing appropriate values into the port output modification set register Pn_OMSR or port output modification clear register Pn_OMCR, respectively. The Pn_OUT.Px bits can also be set, cleared or toggled with register Pn_OMR within the same write operation.

P00_OUT
Port 00 Output Register **(00_H)** **Reset Value: 0000 0000_H**
P02_OUT
Port 02 Output Register **(00_H)** **Reset Value: 0000 0000_H**
Pn_OUT (n=10-11)
Port n Output Register **($F003\ A600_H + n*100_H$)** **Reset Value: 0000 0000_H**
Pn_OUT (n=13-15)
Port n Output Register **($F003\ A600_H + n*100_H$)** **Reset Value: 0000 0000_H**
Pn_OUT (n=20-23)
Port n Output Register **($F003\ AC00_H + n*100_H$)** **Reset Value: 0000 0000_H**
Pn_OUT (n=33-34)
Port n Output Register **($F003\ B200_H + n*100_H$)** **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
Px **(x = 0-15)**	x	rwh	**Port n Output Bit x** This bit determines the level at the output pin Pn.x if the output is selected as GPIO output. 0_B The output level of Pn.x is 0. 1_B The output level of Pn.x is 1. Pn.x can also be set or cleared by control bits of the Pn_OMSR, Pn_OMCR or Pn_OMR registers.

Field	Bits	Type	Description
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

14.3.7 Port Output Modification Register

The port output modification register contains control bits that make it possible to individually set, clear or toggle the logic state of a single port line by manipulating the output register.

P00_OMR
Port 00 Output Modification Register (04_H) Reset Value: 0000 0000_H
P02_OMR
Port 02 Output Modification Register (04_H) Reset Value: 0000 0000_H
Pn_OMR (n=10-11)
Port n Output Modification Register ($F003\ A604_H + n*100_H$) Reset Value: 0000 0000_H
Pn_OMR (n=13-15)
Port n Output Modification Register ($F003\ A604_H + n*100_H$) Reset Value: 0000 0000_H
Pn_OMR (n=20-23)
Port n Output Modification Register ($F003\ AC04_H + n*100_H$) Reset Value: 0000 0000_H
Pn_OMR (n=33-34)
Port n Output Modification Register ($F003\ B204_H + n*100_H$) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCL 15	**PCL 14**	**PCL 13**	**PCL 12**	**PCL 11**	**PCL 10**	**PCL 9**	**PCL 8**	**PCL 7**	**PCL 6**	**PCL 5**	**PCL 4**	**PCL 3**	**PCL 2**	**PCL 1**	**PCL 0**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS 15	**PS 14**	**PS 13**	**PS 12**	**PS 11**	**PS 10**	**PS 9**	**PS 8**	**PS 7**	**PS 6**	**PS 5**	**PS 4**	**PS 3**	**PS 2**	**PS 1**	**PS 0**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
PSx (x = 0-15)	x	w	**Port n Set Bit x** Setting this bit will set or toggle the corresponding bit in the port output register Pn_OUT. Read as 0. The function of this bit is shown in **Table 14-7**. 0_B No operation 1_B Sets or toggles Pn_OUT.Px.

Field	Bits	Type	Description
PCLx (x = 0-15)	x + 16	w	**Port n Clear Bit x** Setting this bit will clear or toggle the corresponding bit in the port output register Pn_OUT. Read as 0. The function of this bit is shown in **Table 14-7**. 0_B No operation 1_B Clears or toggles Pn_OUT.Px.

Table 14-7 Function of the Bits PCLx and PSx

PCLx	PSx	Function
0	0	Bit Pn_OUT.Px is not changed.
0	1	Bit Pn_OUT.Px is set.
1	0	Bit Pn_OUT.Px is reset.
1	1	Bit Pn_OUT.Px is toggled.

14.3.8 Port Output Modification Set Register

The port output modification set register contains control bits that make it possible to individually set the logic state of a single port line by manipulating the output register.

P00_OMSR
Port 00 Output Modification Set Register (90_H) **Reset Value: 0000 0000_H**
P02_OMSR
Port 02 Output Modification Set Register (90_H) **Reset Value: 0000 0000_H**
Pn_OMSR (n=10-11)
Port n Output Modification Set Register ($F003\ A690_H + n*100_H$) **Reset Value: 0000 0000_H**
Pn_OMSR (n=13-15)
Port n Output Modification Set Register ($F003\ A690_H + n*100_H$) **Reset Value: 0000 0000_H**
Pn_OMSR (n=20-23)
Port n Output Modification Set Register ($F003\ AC90_H + n*100_H$) **Reset Value: 0000 0000_H**
Pn_OMSR (n=33-34)
Port n Output Modification Set Register ($F003\ B290_H + n*100_H$) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS 15	PS 14	PS 13	PS 12	PS 11	PS 10	PS 9	PS 8	PS 7	PS 6	PS 5	PS 4	PS 3	PS 2	PS 1	PS 0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
PSx (x = 0-15)	x	w	**Port n Set Bit x** Setting this bit will set the corresponding bit in the port output register Pn_OUT. Read as 0. 0_B No operation 1_B Sets Pn_OUT.Px
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

14.3.9 Port Output Modification Set Registers x

The port output modification set register x, (x = 0, 4, 8, 12) contains control bits to individually set the logic state of a single port line by manipulating the output register.

Register Pn_OMSR0 sets the logic state of Pn.[3:0] port lines

Register Pn_OMSR4 sets the logic state of Pn.[7:4] port lines

Register Pn_OMSR8 sets the logic state of Pn.[11:8] port lines

Register Pn_OMSR12 sets the logic state of Pn.[15:12] port lines

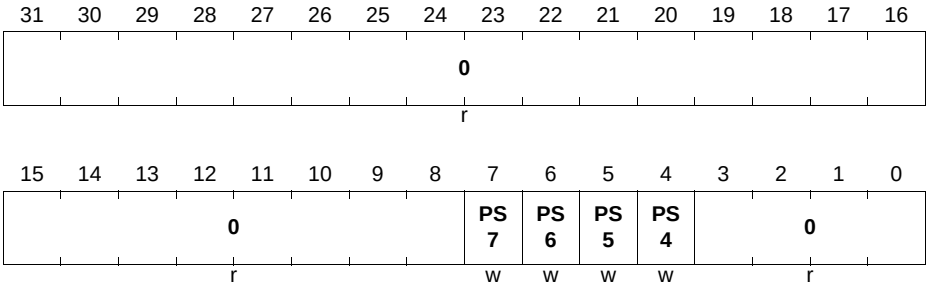
P00_OMSR0
Port 00 Output Modification Set Register 0(70_H) Reset Value: 0000 0000_H
P02_OMSR0
Port 02 Output Modification Set Register 0(70_H) Reset Value: 0000 0000_H
Pn_OMSR0 (n=10-11)
Port n Output Modification Set Register 0(F003 $A670_H$ + n*100_H) Reset Value: 0000 0000_H
Pn_OMSR0 (n=13-15)
Port n Output Modification Set Register 0(F003 $A670_H$ + n*100_H) Reset Value: 0000 0000_H
Pn_OMSR0 (n=20-23)
Port n Output Modification Set Register 0(F003 $AC70_H$ + n*100_H) Reset Value: 0000 0000_H
Pn_OMSR0 (n=33-34)
Port n Output Modification Set Register 0(F003 $B270_H$ + n*100_H) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												PS3	PS2	PS1	PS0
r												w	w	w	w

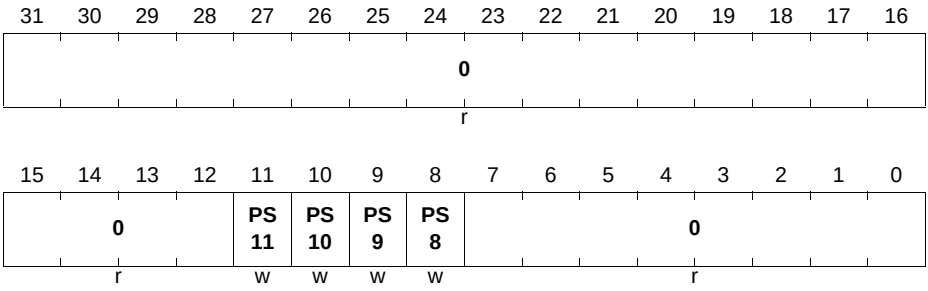
Field	Bits	Type	Description
PSx (x = 0-3)	x	w	**Port n Set Bit x** Setting this bit will set the corresponding bit in the port output register Pn_OUT. Read as 0. 0_B No operation 1_B Sets Pn_OUT.Px
0	[31:4]	r	**Reserved** Read as 0; should be written with 0.

P00_OMSR4
Port 00 Output Modification Set Register 4(74_H) **Reset Value: 0000 0000_H**
P02_OMSR4
Port 02 Output Modification Set Register 4(74_H) **Reset Value: 0000 0000_H**
Pn_OMSR4 (n=10-11)
Port n Output Modification Set Register 4($F003\ A674_H + n*100_H$) **Reset Value: 0000 0000_H**
Pn_OMSR4 (n=14-15)
Port n Output Modification Set Register 4($F003\ A674_H + n*100_H$) **Reset Value: 0000 0000_H**
Pn_OMSR4 (n=20-22)
Port n Output Modification Set Register 4($F003\ AC74_H + n*100_H$) **Reset Value: 0000 0000_H**
P33_OMSR4
Port 33 Output Modification Set Register 4(74_H) **Reset Value: 0000 0000_H**



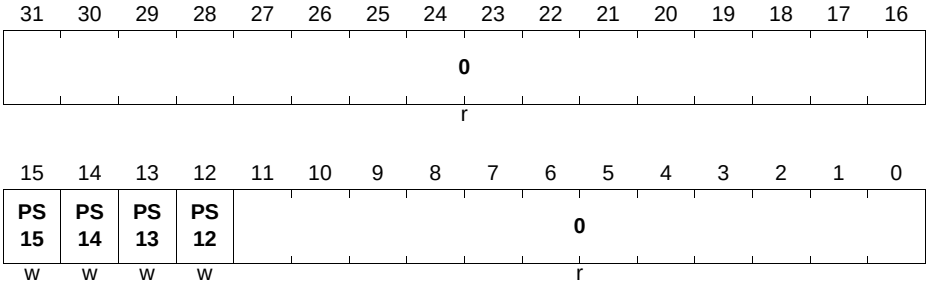
Field	Bits	Type	Description
PSx (x = 7-4)	x	w	**Port n Set Bit x** Setting this bit will set the corresponding bit in the port output register Pn_OUT. Read as 0. 0_B No operation 1_B Sets Pn_OUT.Px
0	[31:8], [3:0]	r	**Reserved** Read as 0; should be written with 0.

P00_OMSR8
Port 00 Output Modification Set Register 8(78_H) Reset Value: 0000 0000_H
P02_OMSR8
Port 02 Output Modification Set Register 8(78_H) Reset Value: 0000 0000_H
P11_OMSR8
Port 11 Output Modification Set Register 8(78_H) Reset Value: 0000 0000_H
Pn_OMSR8 (n=14-15)
Port n Output Modification Set Register 8($F003\ A678_H$ + n*100_H) Reset Value: 0000 0000_H
P20_OMSR8
Port 20 Output Modification Set Register 8(78_H) Reset Value: 0000 0000_H
P33_OMSR8
Port 33 Output Modification Set Register 8(78_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
PSx (x = 11-8)	x	w	**Port n Set Bit x** Setting this bit will set the corresponding bit in the port output register Pn_OUT. Read as 0. 0_B No operation 1_B Sets Pn_OUT.Px
0	[31:12], [7:0]	r	**Reserved** Read as 0; should be written with 0.

P00_OMSR12
Port 00 Output Modification Set Register 12($7C_H$) **Reset Value: 0000 0000$_H$**
P11_OMSR12
Port 11 Output Modification Set Register 12($7C_H$) **Reset Value: 0000 0000$_H$**
P20_OMSR12
Port 20 Output Modification Set Register 12($7C_H$) **Reset Value: 0000 0000$_H$**
P33_OMSR12
Port 33 Output Modification Set Register 12($7C_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
PSx (x = 15-12)	x	w	**Port n Set Bit x** Setting this bit will set the corresponding bit in the port output register Pn_OUT. Read as 0. 0_B No operation 1_B Sets Pn_OUT.Px
0	[31:16], [11:0]	r	**Reserved** Read as 0; should be written with 0.

14.3.10 Port Output Modification Clear Register

The port output modification clear register contains control bits that make it possible to individually clear the logic state of a single port line by manipulating the output register.

P00_OMCR
Port 00 Output Modification Clear Register (94_H) **Reset Value: 0000 0000_H**
P02_OMCR
Port 02 Output Modification Clear Register (94_H) **Reset Value: 0000 0000_H**
Pn_OMCR (n=10-11)
Port n Output Modification Clear Register ($F003\ A694_H + n*100_H$) **Reset Value: 0000 0000_H**
Pn_OMCR (n=13-15)
Port n Output Modification Clear Register ($F003\ A694_H + n*100_H$) **Reset Value: 0000 0000_H**
Pn_OMCR (n=20-23)
Port n Output Modification Clear Register ($F003\ AC94_H + n*100_H$) **Reset Value: 0000 0000_H**
Pn_OMCR (n=33-34)
Port n Output Modification Clear Register ($F003\ B294_H + n*100_H$) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCL 15	**PCL 14**	**PCL 13**	**PCL 12**	**PCL 11**	**PCL 10**	**PCL 9**	**PCL 8**	**PCL 7**	**PCL 6**	**PCL 5**	**PCL 4**	**PCL 3**	**PCL 2**	**PCL 1**	**PCL 0**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
PCLx (x = 0-15)	x + 16	w	**Port n Clear Bit x** Setting this bit will clear the corresponding bit in the port output register Pn_OUT. Read as 0. 0_B No operation 1_B Clears Pn_OUT.Px.
0	[15:0]	r	**Reserved** Read as 0; should be written with 0

14.3.11 Port Output Modification Clear Registers x

The port output modification clear register x, (x = 0, 4, 8, 12) contains control bits to individually clear the logic state of a single port line by manipulating the output register.

Register Pn_OMCR0 clears the logic state of Pn.[3:0] port lines

Register Pn_OMCR4 clears the logic state of Pn.[7:4] port lines

Register Pn_OMCR8 clears the logic state of Pn.[11:8] port lines

Register Pn_OMCR12 clears the logic state of Pn.[15:12] port lines

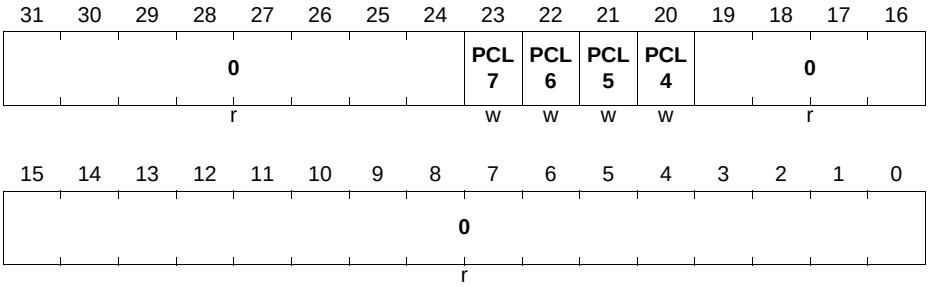
P00_OMCR0
Port 00 Output Modification Clear Register 0(80_H) Reset Value: 0000 0000_H
P02_OMCR0
Port 02 Output Modification Clear Register 0(80_H) Reset Value: 0000 0000_H
Pn_OMCR0 (n=10-11)
Port n Output Modification Clear Register 0(F003 $A680_H$ + n*100_H) Reset Value: 0000 0000_H
Pn_OMCR0 (n=13-15)
Port n Output Modification Clear Register 0(F003 $A680_H$ + n*100_H) Reset Value: 0000 0000_H
Pn_OMCR0 (n=20-23)
Port n Output Modification Clear Register 0(F003 $AC80_H$ + n*100_H) Reset Value: 0000 0000_H
Pn_OMCR0 (n=33-34)
Port n Output Modification Clear Register 0(F003 $B280_H$ + n*100_H) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0												PCL3	PCL2	PCL1	PCL0
r												w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

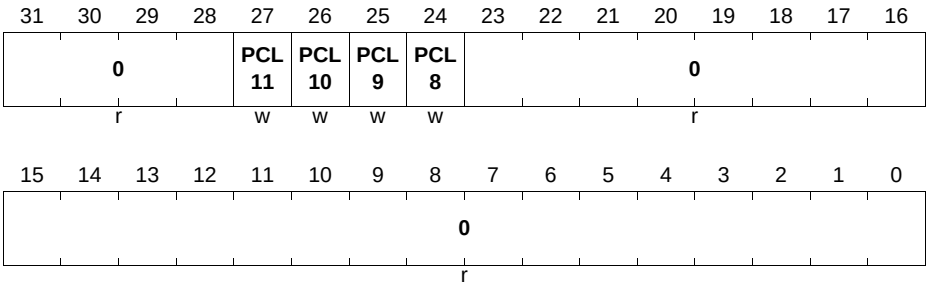
Field	Bits	Type	Description
PCLx (x = 0-3)	x + 16	w	**Port n Clear Bit x** Setting this bit will clear the corresponding bit in the port output register Pn_OUT. Read as 0. 0_B No operation 1_B Clears Pn_OUT.Px
0	[31:20], [15:0]	r	**Reserved** Read as 0; should be written with 0.

P00_OMCR4
Port 00 Output Modification Clear Register 4(84_H) **Reset Value: 0000 0000_H**
P02_OMCR4
Port 02 Output Modification Clear Register 4(84_H) **Reset Value: 0000 0000_H**
Pn_OMCR4 (n=10-11)
Port n Output Modification Clear Register 4($F003\ A684_H + n*100_H$) **Reset Value: 0000 0000_H**
Pn_OMCR4 (n=14-15)
Port n Output Modification Clear Register 4($F003\ A684_H + n*100_H$) **Reset Value: 0000 0000_H**
Pn_OMCR4 (n=20-22)
Port n Output Modification Clear Register 4($F003\ AC84_H + n*100_H$) **Reset Value: 0000 0000_H**
P33_OMCR4
Port 33 Output Modification Clear Register 4(84_H) **Reset Value: 0000 0000_H**



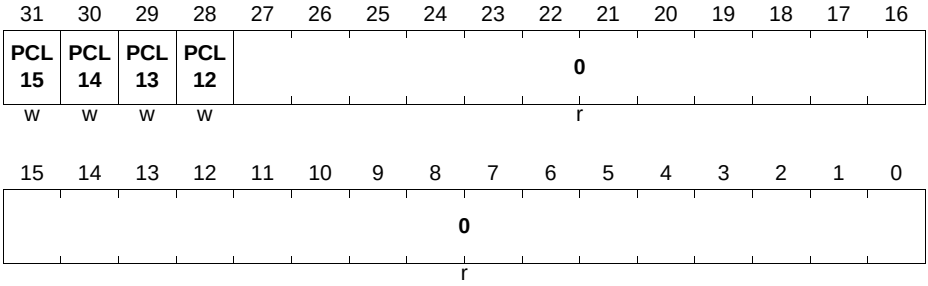
Field	Bits	Type	Description
PCLx (x = 7-4)	x + 16	w	**Port n Clear Bit x** Setting this bit will clear the corresponding bit in the port output register Pn_OUT. Read as 0. 0_B No operation 1_B Clears Pn_OUT.Px
0	[31:24], [19:0]	r	**Reserved** Read as 0; should be written with 0.

P00_OMCR8
Port 00 Output Modification Clear Register 8(88_H) Reset Value: 0000 0000_H
P02_OMCR8
Port 02 Output Modification Clear Register 8(88_H) Reset Value: 0000 0000_H
P11_OMCR8
Port 11 Output Modification Clear Register 8(88_H) Reset Value: 0000 0000_H
Pn_OMCR8 (n=14-15)
Port n Output Modification Clear Register 8($F003\ A688_H$ + n*100_H) Reset Value: 0000 0000_H
P20_OMCR8
Port 20 Output Modification Clear Register 8(88_H) Reset Value: 0000 0000_H
P33_OMCR8
Port 33 Output Modification Clear Register 8(88_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
PCLx (x = 11-8)	x + 16	w	**Port n Clear Bit x** Setting this bit will clear the corresponding bit in the port output register Pn_OUT. Read as 0. 0_B No operation 1_B Clears Pn_OUT.Px
0	[31:28], [23:0]	r	**Reserved** Read as 0; should be written with 0.

P00_OMCR12
Port 00 Output Modification Clear Register 12($8C_H$) **Reset Value: 0000 0000$_H$**
P11_OMCR12
Port 11 Output Modification Clear Register 12($8C_H$) **Reset Value: 0000 0000$_H$**
P20_OMCR12
Port 20 Output Modification Clear Register 12($8C_H$) **Reset Value: 0000 0000$_H$**
P33_OMCR12
Port 33 Output Modification Clear Register 12($8C_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
PCLx (x = 15-12)	x + 16	w	**Port n Clear Bit x** Setting this bit will clear the corresponding bit in the port output register Pn_OUT. Read as 0. 0_B No operation 1_B Clears Pn_OUT.Px
0	[27:0]	r	**Reserved** Read as 0; should be written with 0.

14.3.12 Emergency Stop Register

All digital GPIO lines have an emergency stop logic implemented (see **Figure 14-1**).

Each of these GPIO lines has its own emergency stop enable bit ENx that is located in the emergency stop register Pn_ESR of Port n. If the emergency stop signal becomes active, one of two states can be selected:

- Emergency stop function disabled (ENx = 0):
  The output line remains connected (alternate function).
- Emergency stop function enabled (ENx = 1):
  The mapped output function is disconnected and the safe state is entered by switching to the reset state, input function in tri-state mode or input pull-up. (the content of the corresponding PCx bit fields in register Pn_IOCR will not be considered).

P00_ESR
Port 00 Emergency Stop Register (50_H) Reset Value: 0000 0000_H
P02_ESR
Port 02 Emergency Stop Register (50_H) Reset Value: 0000 0000_H
Pn_ESR (n=10-11)
Port n Emergency Stop Register ($F003\ A650_H + n*100_H$) Reset Value: 0000 0000_H
Pn_ESR (n=13-15)
Port n Emergency Stop Register ($F003\ A650_H + n*100_H$) Reset Value: 0000 0000_H
Pn_ESR (n=20-23)
Port n Emergency Stop Register ($F003\ AC50_H + n*100_H$) Reset Value: 0000 0000_H
Pn_ESR (n=33-34)
Port n Emergency Stop Register ($F003\ B250_H + n*100_H$) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN 15	EN 14	EN 13	EN 12	EN 11	EN 10	EN 9	EN 8	EN 7	EN 6	EN 5	EN 4	EN 3	EN 2	EN 1	EN 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENx (x = 0-15)	x	rw	**Emergency Stop Enable for Port n Pin x** This bit enables the emergency stop function for all GPIO lines. If the emergency stop condition is met and enabled, the output selection is automatically switched from alternate output function to GPIO input function with tri-state. See **Page 14-38**. 0_B Emergency stop function for Pn.x is disabled. 1_B Emergency stop function for Pn.x is enabled.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

14.3.13 Port Input Register

The logic level of a GPIO pin can be read via the read-only port input register Pn_IN. Reading the Pn_IN register always returns the current logical value at the GPIO pin independently whether the pin is selected as input or output.

P00_IN
Port 00 Input Register **(24_H)** **Reset Value: 0000 XXXX$_H$**
P02_IN
Port 02 Input Register **(24_H)** **Reset Value: 0000 XXXX$_H$**
Pn_IN (n=10-11)
Port n Input Register **(F003 A624$_H$ + n*100$_H$)** **Reset Value: 0000 XXXX$_H$**
Pn_IN (n=13-15)
Port n Input Register **(F003 A624$_H$ + n*100$_H$)** **Reset Value: 0000 XXXX$_H$**
Pn_IN (n=20-23)
Port n Input Register **(F003 AC24$_H$ + n*100$_H$)** **Reset Value: 0000 XXXX$_H$**
Pn_IN (n=33-34)
Port n Input Register **(F003 B224$_H$ + n*100$_H$)** **Reset Value: 0000 XXXX$_H$**
P40_IN
Port 40 Input Register **(24_H)** **Reset Value: 0000 XXXX$_H$**
P41_IN
Port 41 Input Register **(24_H)** **Reset Value: 0000 XXXX$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	**P14**	**P13**	**P12**	**P11**	**P10**	**P9**	**P8**	**P7**	**P6**	**P5**	**P4**	**P3**	**P2**	**P1**	**P0**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
Px **(x = 0-15)**	x	rh	**Port n Input Bit x** This bit indicates the level at the input pin Pn.x. 0_B The input level of Pn.x is 0. 1_B The input level of Pn.x is 1.
0	[31:16]	r	**Reserved** Read as 0.

14.3.14 Access Protection Registers

The Access Enable Register 0 controls write[1)] access for transactions with the on chip bus master TAG ID 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 and ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000_B, EN1 -> TAG ID 000001_B , ... ,EN31 -> TAG ID 011111_B.

Each port has its own dedicated ACCEN0 and ACCEN1 registers.

P00_ACCEN0
Port 00 Access Enable Register 0 (FC_H) Reset Value: FFFF FFFF$_H$
P02_ACCEN0
Port 02 Access Enable Register 0 (FC_H) Reset Value: FFFF FFFF$_H$
Px_ACCEN0 (x=10-11)
Port x Access Enable Register 0($F003\ A6FC_H + x*100_H$) Reset Value: FFFF FFFF$_H$
Px_ACCEN0 (x=13-15)
Port x Access Enable Register 0($F003\ A6FC_H + x*100_H$) Reset Value: FFFF FFFF$_H$
Px_ACCEN0 (x=20-23)
Port x Access Enable Register 0($F003\ ACFC_H + x*100_H$) Reset Value: FFFF FFFF$_H$
Px_ACCEN0 (x=33-34)
Port x Access Enable Register 0($F003\ B2FC_H + x*100_H$) Reset Value: FFFF FFFF$_H$
P40_ACCEN0
Port 40 Access Enable Register 0 (FC_H) Reset Value: FFFF FFFF$_H$
P41_ACCEN0
Port 41 Access Enable Register 0 (FC_H) Reset Value: FFFF FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN 31	**EN 30**	**EN 29**	**EN 28**	**EN 27**	**EN 26**	**EN 25**	**EN 24**	**EN 23**	**EN 22**	**EN 21**	**EN 20**	**EN 19**	**EN 18**	**EN 17**	**EN 16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN 15	**EN 14**	**EN 13**	**EN 12**	**EN 11**	**EN 10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

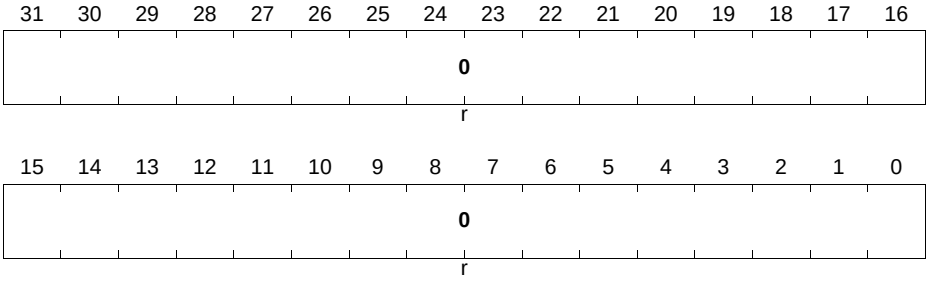
1) The BPI_FPI Access Enable functionality controls only write transactions to the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers.

Field	Bits	Type	Description
ENx (x = 0-31)	x	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

The Access Enable Register 1 controls write[1)] access for transactions with the on chip bus master TAG ID 100000_B to 111111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the AURIX devices.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001B, ... ,EN31 -> TAG ID 111111B.

P00_ACCEN1
Port 00 Access Enable Register 1 ($F8_H$) Reset Value: 0000 0000$_H$
P02_ACCEN1
Port 02 Access Enable Register 1 ($F8_H$) Reset Value: 0000 0000$_H$
Px_ACCEN1 (x=10-11)
Port x Access Enable Register 1($F003\ A6F8_H + x*100_H$) Reset Value: 0000 0000$_H$
Px_ACCEN1 (x=13-15)
Port x Access Enable Register 1($F003\ A6F8_H + x*100_H$) Reset Value: 0000 0000$_H$
Px_ACCEN1 (x=20-23)
Port x Access Enable Register 1($F003\ ACF8_H + x*100_H$) Reset Value: 0000 0000$_H$
Px_ACCEN1 (x=33-34)
Port x Access Enable Register 1($F003\ B2F8_H + x*100_H$) Reset Value: 0000 0000$_H$
P40_ACCEN1
Port 40 Access Enable Register 1 ($F8_H$) Reset Value: 0000 0000$_H$
P41_ACCEN1
Port 41 Access Enable Register 1 ($F8_H$) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
0	[31:0]	r	**Reserved** Read as 0; should be written with 0.

14.4 Port 00

This section describes the Port 00 functionality.

14.4.1 Port 00 Registers

The following registers are available on Port 00:

Table 14-8 Port 00 Registers

Register Short Name	Register Long Name	Offset[1] Address	Description see
P00_OUT	Port 00 Output Register	0000_H	Page 14-46[2]
P00_OMR	Port 00 Output Modification Register	0004_H	Page 14-46[2]
P00_IOCR0	Port 00 Input/Output Control Register 0	0010_H	Page 14-14
P00_IOCR4	Port 00 Input/Output Control Register 4	0014_H	Page 14-15
P00_IOCR8	Port 00 Input/Output Control Register 8	0018_H	Page 14-16
P00_IOCR12	Port 00 Input/Output Control Register 12	$001C_H$	Page 14-47[2]
P00_IN	Port 00 Input Register	0024_H	Page 14-47[2]
P00_PDR0	Port 00 Pad Driver Mode 0 Register	0040_H	Page 14-20
P00_PDR1	Port 00 Pad Driver Mode 1 Register	0044_H	Page 14-48[2]
P00_ESR	Port 00 Emergency Stop Register	0050_H	Page 14-48[2]
P00_OMSR0	Port 00 Output Modification Set Register 0	0070_H	Page 14-29
P00_OMSR4	Port 00 Output Modification Set Register 4	0074_H	Page 14-30
P00_OMSR8	Port 00 Output Modification Set Register 8	0078_H	Page 14-31
P00_OMSR12	Port 00 Output Modification Set Register 12	$007C_H$	Page 14-46[2]
P00_OMCR0	Port 00 Output Modification Clear Register 0	0080_H	Page 14-34
P00_OMCR4	Port 00 Output Modification Clear Register 4	0084_H	Page 14-35
P00_OMCR8	Port 00 Output Modification Clear Register 8	0088_H	Page 14-36
P00_OMCR12	Port 00 Output Modification Clear Register 12	$008C_H$	Page 14-46[2]
P00_OMSR	Port 00 Output Modification Set Register	0090_H	Page 14-46[2]
P00_OMCR	Port 00 Output Modification Clear Register	0094_H	Page 14-46[2]
P00_ACCEN1	Port 00 Access Enable Register 1	$00F8_H$	Page 14-43
P00_ACCEN0	Port 00 Access Enable Register 0	$00FC_H$	Page 14-41

1) The absolute addresses are calculated by adding the offset address to the module base address (see Table 14-3)

2) These registers are listed and noted here in the Port 00 section because they differ from the general port register description given in Section 14.3.

14.4.1.1 Port 00 Output Register

The basic P00_OUT register functionality is described on **Page 14-24**. Port lines P00.[15:13] are not connected. Reading the P00_OUT bits P[15:13] returns the value that was last written. These connected bits can be also set/reset by the corresponding bits in P00_OMR, P00_OMSR, P00_OMCR, P00_OMSRx (x=0,4,8,12), P00_OMCRx (x=0,4,8,12).

14.4.1.2 Port 00 Output Modification Register

The basic P00_OMR register functionality is described on **Page 14-26**. However, port lines P00.[15:13] are not connected. The P00_OMR bits PS[15:13] and PCL[15:13] have no direct effect on port lines but only on register bits P00_OUT.P[15:13].

14.4.1.3 Port 00 Output Modification Set Register

The basic P00_OMSR register functionality is described on **Page 14-28**. However, port lines P00.[15:13] are not connected. The P00_OMSR bits PS[15:13] have no direct effect on port lines but only on register bits P00_OUT.P[15:13].

14.4.1.4 Port 00 Output Modification Set Register 12

The basic P00_OMSR12 register functionality is described on **Page 14-32**. However, port lines P00.[15:13] are not connected. The P00_OMSR12 bits PS[15:13] have no direct effect on port lines but only on register bits P00_OUT.P[15:13].

14.4.1.5 Port 00 Output Modification Clear Register

The basic P00_OMCR register functionality is described on **Page 14-33**. However, port lines P00.[15:13] are not connected. The P00_OMCR bits PCL[15:13] have no direct effect on port lines but only on register bits P00_OUT.P[15:13].

14.4.1.6 Port 00 Output Modification Clear Register 12

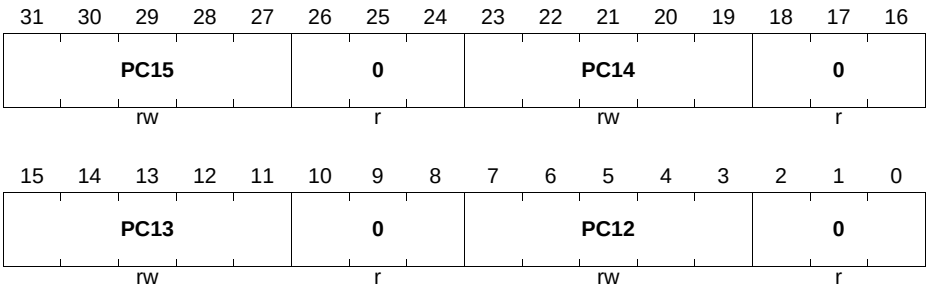
The basic P00_OMCR12 register functionality is described on **Page 14-37**. However, port lines P00.[15:13] are not connected. The P00_OMCR12 bits PCL[15:13] have no direct effect on port lines but only on register bits P00_OUT.P[15:13].

14.4.1.7 Port 00 Input/Output Control Register 12

The PC13, PC14 and PC15 bit fields in register P00_IOCR12 are not connected.

P00_IOCR12
Port 00 Input/Output Control Register 12

($1C_H$) **Reset Value: $0000\ 0000_H$**



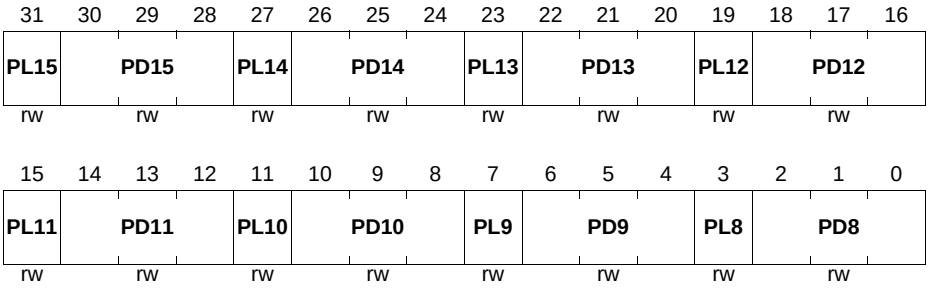
Field	Bits	Type	Description
PC13	[15:11]	rw	**Reserved** Read as 00000_B after reset; returns value that was written.
PC14	[23:19]	rw	**Reserved** Read as 00000_B after reset; returns value that was written.
PC15	[31:27]	rw	**Reserved** Read as 00000_B after reset; returns value that was written.
PC12	[7:3]	rw	**Port Control for Port 00.12** (coding see **Table 14-5**)
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

14.4.1.8 Port 00 Input Register

The basic P00_IN register functionality is described on **Page 14-40**. However, port lines P00.[15:13] are not connected. Therefore, bits P[15:13] in register P00_IN are always read as 0.

14.4.1.9 Port 00 Pad Driver Mode 1 Register

P00_PDR1
Port 00 Pad Driver Mode 1 Register (44$_H$) **Reset Value: 3333 3333$_H$**



Field	Bits	Type	Description
PD8	[2:0]	rw	**Pad Driver Mode for Port 00 Pin 8**
PD9	[6:4]	rw	**Pad Driver Mode for Port 00 Pin 9**
PD10	[10:8]	rw	**Pad Driver Mode for Port 00 Pin 10**
PD11	[14:12]	rw	**Pad Driver Mode for Port 00 Pin 11**
PD12	[18:16]	rw	**Pad Driver Mode for Port 00 Pin 12**
PD13, **PD14,** **PD15**	[22:20], [26:24], [30:28]	rw	**Reserved** Read as 011$_B$ after reset; returns value that was written, must be written with reset value.
PL8, **PL9,** **PL10,** **PL11,** **PL12,** **PL13,** **PL14,** **PL15**	3, 7, 11, 15, 19, 23, 27, 31	rw	**Reserved** Read as 0, returns values last written, must be written with 0.

14.4.1.10 Port 00 Emergency Stop Register

The basic P00_ESR register functionality is described on **Page 14-38**. Port lines P00.[15:13] are not connected. Reading the P00_ESR bits EN[15:13] returns the value that was last written.

14.5 Port 02

This section describes the Port 02 functionality.

14.5.1 Port 02 Registers

The following registers are available on Port 02:

Table 14-9 Port 02 Registers

Register Short Name	Register Long Name	Offset[1] Address	Description see
P02_OUT	Port 02 Output Register	0000_H	Page 14-51 [2]
P02_OMR	Port 02 Output Modification Register	0004_H	Page 14-51 [2]
P02_IOCR0	Port 02 Input/Output Control Register 0	0010_H	Page 14-14
P02_IOCR4	Port 02 Input/Output Control Register 4	0014_H	Page 14-15
P02_IOCR8	Port 02 Input/Output Control Register 8	0018_H	Page 14-52 [2]
P02_IN	Port 02 Input Register	0024_H	Page 14-52 [2]
P02_PDR0	Port 02 Pad Driver Mode 0 Register	0040_H	Page 14-20
P02_PDR1	Port 02 Pad Driver Mode 1 Register	0044_H	Page 14-53 [2]
P02_ESR	Port 02 Emergency Stop Register	0050_H	Page 14-53 [2]
P02_OMSR0	Port 02 Output Modification Set Register 0	0070_H	Page 14-29
P02_OMSR4	Port 02 Output Modification Set Register 4	0074_H	Page 14-30
P02_OMSR8	Port 02 Output Modification Set Register 8	0078_H	Page 14-51 [2]
P02_OMCR0	Port 02 Output Modification Clear Register 0	0080_H	Page 14-34
P02_OMCR4	Port 02 Output Modification Clear Register 4	0084_H	Page 14-35
P02_OMCR8	Port 02 Output Modification Clear Register 8	0088_H	Page 14-51 [2]
P02_OMSR	Port 02 Output Modification Set Register	0090_H	Page 14-51 [2]
P02_OMCR	Port 02 Output Modification Clear Register	0094_H	Page 14-51 [2]
P02_ACCEN1	Port 02 Access Enable Register 1	$00F8_H$	Page 14-43
P02_ACCEN0	Port 02 Access Enable Register 0	$00FC_H$	Page 14-41

1) The absolute addresses are calculated by adding the offset address to the module base address (see **Table 14-3**)
2) These registers are listed and noted here in the Port 02 section because they differ from the general port register description given in **Section 14.3**.

14.5.1.1 Port 02 Output Register

The basic P02_OUT register functionality is described on **Page 14-24**. Port lines P02.[15:9] are not connected. Reading the P02_OUT bits P[15:12] always return 0. Reading P[11:9] returns the value that was last written. These connected bits can be also set/reset by the corresponding bits in P02_OMR, P02_OMSR, P02_OMCR, P02_OMSRx (x=0,4,8), P02_OMCRx (x=0,4,8).

14.5.1.2 Port 02 Output Modification Register

The basic P02_OMR register functionality is described on **Page 14-26**. However, port lines P02.[15:9] are not connected. The P02_OMR bits PS[11:9] and PCL[11:9] have no direct effect on port lines but only on register bits P02_OUT.P[11:9].

14.5.1.3 Port 02 Output Modification Set Register

The basic P02_OMSR register functionality is described on **Page 14-28**. However, port lines P02.[15:9] are not connected. The P02_OMSR bits PS[11:9] have no direct effect on port lines but only on register bits P02_OUT.P[11:9].

14.5.1.4 Port 02 Output Modification Set Register 8

The basic P02_OMSR8 register functionality is described on **Page 14-31**. However, port lines P02.[11:9] are not connected. The P02_OMSR8 bits PS[11:9] have no direct effect on port lines but only on register bits P02_OUT.P[11:9].

14.5.1.5 Port 02 Output Modification Clear Register

The basic P02_OMCR register functionality is described on **Page 14-33**. However, port lines P02.[15:9] are not connected. The P02_OMCR bits PCL[11:9] have no direct effect on port lines but only on register bits P02_OUT.P[11:9].

14.5.1.6 Port 02 Output Modification Clear Register 8

The basic P02_OMCR8 register functionality is described on **Page 14-36**. However, port lines P02.[11:9] are not connected. The P02_OMCR8 bits PCL[11:9] have no direct effect on port lines but only on register bits P02_OUT.P[11:9].

14.5.1.7 Port 02 Input/Output Control Register 8

The PC[11:9] bit fields in register P02_IOCR8 are not connected.

P02_IOCR8
Port 02 Input/Output Control Register 8

(18_H) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PC11					0			PC10					0		
rw					r			rw					r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC9					0			PC8					0		
rw					r			rw					r		

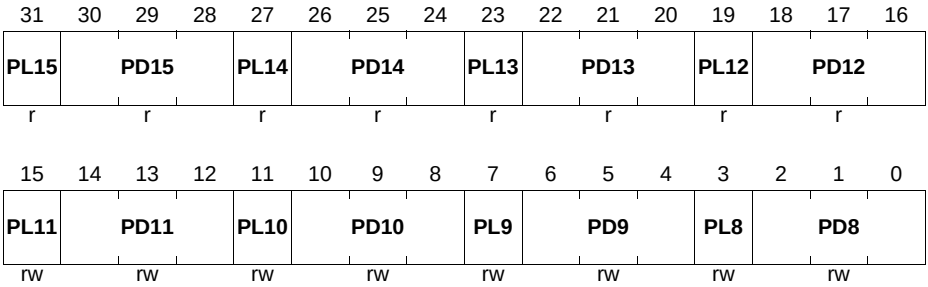
Field	Bits	Type	Description
PC8	[7:3]	rw	**Port Control for Port 02 Pin 8** Coding see **Table 14-5**.
PC9, PC10, PC11	[15:11], [23:19], [31:27]	rw	**Reserved** Read as 00000_B after reset; returns value that was written.
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

14.5.1.8 Port 02 Input Register

The basic P02_IN register functionality is described on **Page 14-40**. However, port lines P02.[15:9] are not connected. Therefore, bits P[15:9] in register P02_IN are always read as 0.

14.5.1.9 P02 Pad Driver Mode 1 Register

P02_PDR1
Port 02 Pad Driver Mode 1 Register (44_H) **Reset Value: $0000\ 3333_H$**



Field	Bits	Type	Description
PD8	[2:0]	rw	**Pad Driver Mode for Port n Pin 8**
PD9, PD10, PD11	[6:4], [10:8], [14:12]	rw	**Reserved** Read as 011_B after reset; returns value that was written, must be written with reset value.
PD12, PD13, PD14, PD15	[18:16], [22:20], [26:24], [30:28]	r	**Reserved** Read as 000_B after reset; should be written as 0.
PL8, PL9, PL10, PL11	3, 7, 11, 15	rw	**Reserved** Read as 0_B after reset; returns value that was written, have to be written with 0.
PL12, PL13, PL14, PL15	19, 23, 27, 31	r	**Reserved** Read as 0_B after reset; should be written as 0.

14.5.1.10 Port 02 Emergency Stop Register

The basic P02_ESR register functionality is described on **Page 14-38**. Port lines P02.[15:9] are not connected. Reading the P02_ESR bits EN[15:12] always returns 0. Reading the P02_ESR bits EN[11:9] returns value that was last written.

14.6 Port 10

This section describes the Port 10 functionality.

14.6.1 Port 10 Registers

The following registers are available on Port 10:

Table 14-10 Port 10 Registers

Register Short Name	Register Long Name	Offset[1] Address	Description see
P10_OUT	Port 10 Output Register	0000_H	Page 14-55[2]
P10_OMR	Port 10 Output Modification Register	0004_H	Page 14-56[2]
P10_IOCR0	Port 10 Input/Output Control Register 0	0010_H	Page 14-57[2]
P10_IOCR4	Port 10 Input/Output Control Register 4	0014_H	Page 14-58[2]
P10_IN	Port 10 Input Register	0024_H	Page 14-58[2]
P10_PDR0	Port 10 Pad Driver Mode 0 Register	0040_H	Page 14-59[2]
P10_ESR	Port 10 Emergency Stop Register	0050_H	Page 14-59[2]
P10_OMSR0	Port 10 Output Modification Set Register 0	0070_H	Page 14-56[2]
P10_OMSR4	Port 10 Output Modification Set Register 4	0074_H	Page 14-56[2]
P10_OMCR0	Port 10 Output Modification Clear Register 0	0080_H	Page 14-56[2]
P10_OMCR4	Port 10 Output Modification Clear Register 4	0084_H	Page 14-56[2]
P10_OMSR	Port 10 Output Modification Set Register	0090_H	Page 14-56[2]
P10_OMCR	Port 10 Output Modification Clear Register	0094_H	Page 14-56[2]
P10_ACCEN1	Port 10 Access Enable Register 1	$00F8_H$	Page 14-43
P10_ACCEN0	Port 10 Access Enable Register 0	$00FC_H$	Page 14-41

1) The absolute addresses are calculated by adding the offset address to the module base address (see Table 14-3)

2) These registers are listed and noted here in the Port 10 section because they differ from the general port register description given in Section 14.3.

14.6.1.1 Port 10 Output Register

The basic P10_OUT register functionality is described on Page 14-24. Port lines P10.0, P10.4, P10.[15:7] are not connected. Reading the P10_OUT bits P10.0, P10.4, P10.7 returns the value that was last written. Reading P10.[15:8] returns 0. These connected bits can be also set/reset by the corresponding bits in P10_OMR, P10_OMSR, P10_OMCR, P10_OMSRx (x=0,4), P10_OMCRx (x=0,4).

14.6.1.2 Port 10 Output Modification Register

The basic P10_OMR register functionality is described on **Page 14-26**. However, port lines P10.0, P10.4, P10.[15:7] are not connected. The P10_OMR bits PS0, PS4, PS[15:7] and PCL0, PCL4, PCL7 have no direct effect on port lines but only on register bits P10_OUT.P0, P4 and P7.

14.6.1.3 Port 10 Output Modification Set Register

The basic P10_OMSR register functionality is described on **Page 14-28**. However, port lines P10.0, P10.4, P10.[15:7] are not connected. The P10_OMSR bits PS0, PS4, PS7 have no direct effect on port lines but only on register bits P10_OUT.P0, P4 and P7.

14.6.1.4 Port 10 Output Modification Set Register 0

The basic P10_OMSR0 register functionality is described on **Page 14-29**. However, port lines P10.0 is not connected. The P10_OMSR0 bits PS0 has no direct effect on port lines but only on register bits P10_OUT.P0.

14.6.1.5 Port 10 Output Modification Set Register 4

The basic P10_OMSR4 register functionality is described on **Page 14-30**. However, port lines P10.4 and P10.7 are not connected. The P10_OMSR4 bits PS4 and PS7 have no direct effect on port lines but only on register bits P10_OUT.P4 and P7.

14.6.1.6 Port 10 Output Modification Clear Register

The basic P10_OMCR register functionality is described on **Page 14-33**. However, port lines P10.0, P10.4, P10.7 are not connected. The P10_OMCR bits PCL0, PCL4, PCL7 have no direct effect on port lines but only on register bits P10_OUT.P0, P4 and P7.

14.6.1.7 Port 10 Output Modification Clear Register 0

The basic P10_OMCR0 register functionality is described on **Page 14-34**. However, port lines P10.0 is not connected. The P10_OMSR0 bits PCL0 has no direct effect on port lines but only on register bits P10_OUT.P0.

14.6.1.8 Port 10 Output Modification Clear Register 4

The basic P10_OMCR4 register functionality is described on **Page 14-35**. However, port lines P10.4 and P10.7 are not connected. The P10_OMCR4 bits PCL4 and PCL7 have no direct effect on port lines but only on register bits P10_OUT.P4 and P7.

14.6.1.9 Port 10 Input/Output Control Register 0

The PC0 bit field in register P10_IOC04 is not connected.

P10_IOCR0
Port 10 Input/Output Control Register 0

(10_H) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PC3					**0**			**PC2**					**0**		
rw					r			rw					r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC1					**0**			**PC0**					**0**		
rw					r			rw					r		

Field	Bits	Type	Description
PC0	[7:3]	rw	**Reserved** Read as 00000_B after reset; returns value that was written.
PC1, PC2, PC3	[15:11], [23:19], [31:27]	rw	**Port Control for Port 10 Pin 1 to 3** (coding see **Table 14-5**)
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

14.6.1.10 Port 10 Input/Output Control Register 4

The PC4 and PC7 bit fields in register P10_IOCR4 are not connected.

P10_IOCR4
Port 10 Input/Output Control Register 4

(14_H) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PC7					**0**			**PC6**					**0**		
rw					r			rw					r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC5					**0**			**PC4**					**0**		
rw					r			rw					r		

Field	Bits	Type	Description
PC4, PC7	[7:3], [31:27]	rw	**Reserved** Read as 00000_B after reset; returns value that was written.
PC5, PC6	[15:11], [23:19]	rw	**Port Control for Port 10 Pin 5 to 6** (coding see **Table 14-5**)
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

14.6.1.11 Port 10 Input Register

The basic P10_IN register functionality is described on **Page 14-40**. However, port lines P10.0, P10.4, P10.[15:7] are not connected. Therefore, bits P0, P4, P[15:7] in register P10_IN are always read as 0.

14.6.1.12 Port 10 Pad Driver Mode 0 Register

The basic P10_PDR0 register functionality is described on **Page 14-20**.

P10_PDR0
Port 10 Pad Driver Mode 0 Register (40$_H$) **Reset Value: 3333 3333$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PL7		PD7		PL6		PD6		PL5		PD5		PL4		PD4	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PL3		PD3		PL2		PD2		PL1		PD1		PL0		PD0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
PD1	[6:4]	rw	**Pad Driver Mode for Port 20 Pin 1**
PD2	[10:8]	rw	**Pad Driver Mode for Port 20 Pin 2**
PD3	[14:12]	rw	**Pad Driver Mode for Port 20 Pin 3**
PD5	[22:20]	rw	**Pad Driver Mode for Port 20 Pin 5**
PD6	[26:24]	rw	**Pad Driver Mode for Port 20 Pin 6**
PD0, **PD4,** **PD7**	[2:0], [18:16], [30:28]	rw	**Reserved** Read as 011_B after reset; returns value that was written, must be written with reset value.
PL0, **PL1,** **PL2,** **PL3,** **PL4,** **PL5,** **PL6,** **PL7**	3, 7, 11, 15, 19, 23, 27, 31	rw	**Reserved** Read as 0, returns values last written, must be written with 0.

14.6.1.13 Port 10 Emergency Stop Register

The basic P10_ESR register functionality is described on **Page 14-38**. Port lines P10.0, P10.4, P[15:7] are not connected. Reading the P10_ESR bits EN[15:8] always returns 0. Reading the P10_ESR bits EN0, EN4, EN7 returns value that was last written.

14.7 Port 11

This section describes the Port 11 functionality.

14.7.1 Port 11 Registers

The following registers are available on Port 11:

Table 14-11 Port 11 Registers

Register Short Name	Register Long Name	Offset[1] Address	Description see
P11_OUT	Port 11 Output Register	0000_H	Page 14-62[2]
P11_OMR	Port 11 Output Modification Register	0004_H	Page 14-62[2]
P11_IOCR0	Port 11 Input/Output Control Register 0	0010_H	Page 14-64[2]
P11_IOCR4	Port 11 Input/Output Control Register 4	0014_H	Page 14-65[2]
P11_IOCR8	Port 11 Input/Output Control Register 8	0018_H	Page 14-16
P11_IOCR12	Port 11 Input/Output Control Register 12	$001C_H$	Page 14-66[2]
P11_IN	Port 11 Input Register	0024_H	Page 14-66[2]
P11_PDR0	Port 11 Pad Driver Mode 0 Register	0040_H	Page 14-67[2]
P11_PDR1	Port 11 Pad Driver Mode 1 Register	0044_H	Page 14-68[2]
P11_ESR	Port 11 Emergency Stop Register	0050_H	Page 14-68[2]
P11_OMSR0	Port 11 Output Modification Set Register 0	0070_H	Page 14-62[2]
P11_OMSR4	Port 11 Output Modification Set Register 4	0074_H	Page 14-62[2]
P11_OMSR8	Port 11 Output Modification Set Register 8	0078_H	Page 14-31
P11_OMSR12	Port 11 Output Modification Set Register 12	$007C_H$	Page 14-62[2]
P11_OMCR0	Port 11 Output Modification Clear Register 0	0080_H	Page 14-63[2]
P11_OMCR4	Port 11 Output Modification Clear Register 4	0084_H	Page 14-63[2]
P11_OMCR8	Port 11 Output Modification Clear Register 8	0088_H	Page 14-36
P11_OMCR12	Port 11 Output Modification Clear Register 12	$008C_H$	Page 14-63[2]
P11_OMSR	Port 11 Output Modification Set Register	0090_H	Page 14-62[2]
P11_OMCR	Port 11 Output Modification Clear Register	0094_H	Page 14-62[2]
P11_ACCEN1	Port 11 Access Enable Register 1	$00F8_H$	Page 14-43
P11_ACCEN0	Port 11 Access Enable Register 0	$00FC_H$	Page 14-41

1) The absolute addresses are calculated by adding the offset address to the module base address (see Table 14-3)

2) These registers are listed and noted here in the Port 10 section because they differ from the general port register description given in Section 14.3.

14.7.1.1 Port 11 Output Register

The basic P11_OUT register functionality is described on **Page 14-24**. Port lines P11.[15:13], P11.7, P11.[5:4], P11.[1:0] are not connected. Reading the P11_OUT P[15:13], 7, [5:4], [1:0] bits return the value that was last written (0 after reset). These connected bits can be also set/reset by the corresponding bits in P10_OMR, P11_OMSR, P11_OMCR, P11_OMSRx (x=0,4,8,12), P11_OMCRx (x=0,4,8,12).

14.7.1.2 Port 11 Output Modification Register

The basic P11_OMR register functionality is described on **Page 14-26**. However, port lines P11.[15:13], P11.7, P11.[5:4], P11.[1:0] are not connected. The P11_OMR bits PS[15:13], PS7, PS[5:4], PS[1:0] and PCL[15:13], PCL7, PCL[5:4], PCL[1:0] have no direct effect on port lines but only on register bits P11_OUT.P[15:13], P7,P[5:4],P[1:0].

14.7.1.3 Port 11 Output Modification Set Register

The basic P11_OMSR register functionality is described on **Page 14-28**. However, port lines P11.[15:13], P11.7, P11.[5:4], P11.[1:0] are not connected. The P11_OMSR bits PS[15:13], PS7, PS[5:4], PS[1:0] have no direct effect on port lines but only on register bits P11_OUT.P[15:13], P7,P[5:4],P[1:0].

14.7.1.4 Port 11 Output Modification Set Register 0

The basic P11_OMSR0 register functionality is described on **Page 14-29**. However, port lines P11.[1:0] are not connected. The P11_OMSR0 bits PS[1:0] have no direct effect on port lines but only on register bits P11_OUT.P[1:0].

14.7.1.5 Port 11 Output Modification Set Register 4

The basic P11_OMSR4 register functionality is described on **Page 14-30**. However, port lines P11.[5:4], P11.7 are not connected. The P11_OMSR4 bits PS[5:4], PS7 have no direct effect on port lines but only on register bits P11_OUT.P[5:4], P11_OUT.P7.

14.7.1.6 Port 11 Output Modification Set Register 12

The basic P11_OMSR12 register functionality is described on **Page 14-32**. However, port lines P11.[15:13] are not connected. The P11_OMSR12 bits PS[15:13] have no direct effect on port lines but only on register bits P11_OUT.P[15:13].

14.7.1.7 Port 11 Output Modification Clear Register

The basic P11_OMCR register functionality is described on **Page 14-33**. However, port lines P11.[15:13], P11.7, P11.[5:4], P11.[1:0] are not connected. The P11_OMCR bits

PCL[15:13], PCL7, PCL[5:4], PCL[1:0] have no direct effect on port lines but only on register bits P11_OUT.P[15:13], P7,P[5:4],P[1:0].

14.7.1.8 Port 11 Output Modification Clear Register 0

The basic P11_OMCR0 register functionality is described on **Page 14-34**. However, port lines P11.[1:0] are not connected. The P11_OMCR0 bits PCL[1:0] have no direct effect on port lines but only on register bits P11_OUT.P[1:0].

14.7.1.9 Port 11 Output Modification Clear Register 4

The basic P11_OMCR4 register functionality is described on **Page 14-30**. However, port lines P11.[5:4], P11.7 are not connected. The P11_OMCR4 bits PCL[5:4], PCL7 have no direct effect on port lines but only on register bits P11_OUT.P[5:4], P11_OUT.P7.

14.7.1.10 Port 11 Output Modification Clear Register 12

The basic P11_OMCR12 register functionality is described on **Page 14-37**. However, port lines P11.[15:13] are not connected. The P11_OMCR12 bits PCL[15:13] have no direct effect on port lines but only on register bits P11_OUT.P[15:13].

14.7.1.11 Port 11 Input/Output Control Register 0

The PC[1:0] bit fields in register P11_IOCR0 are not connected.

P11_IOCR0
Port 11 Input/Output Control Register 0

(10_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PC3					**0**			**PC2**					**0**		
rw					r			rw					r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC1					**0**			**PC0**					**0**		
rw					r			rw					r		

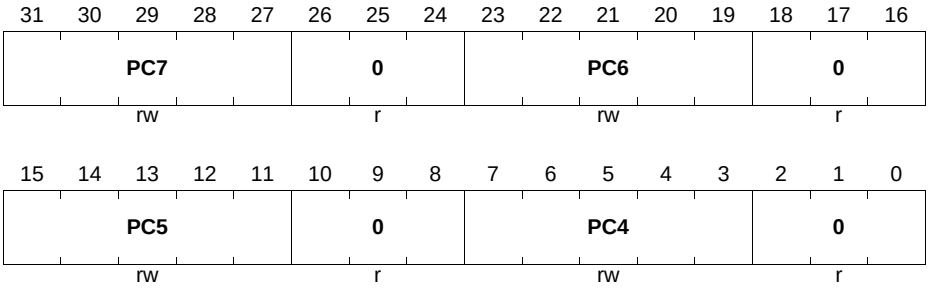
Field	Bits	Type	Description
PC2	[23:19]	rw	**Port Control for Port 11 Pin 2**
PC3	[31:27]	rw	**Port Control for Port 11 Pin 3**
PC0, PC1	[7:3], [15:11]	rw	**Reserved** Read as 00000_B after reset; returns value that was written.
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

14.7.1.12 Port 11 Input/Output Control Register 4

The PC[5:4] and PC7 bit fields in register P11_IOCR4 are not connected.

P11_IOCR4
Port 11 Input/Output Control Register 4

(14_H) **Reset Value: 0000 0000$_H$**



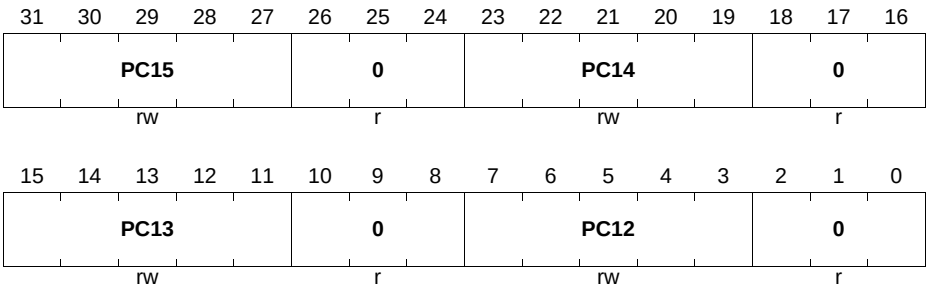
Field	Bits	Type	Description
PC4, PC5, PC7	[7:3], [15:11], [31:27]	rw	**Reserved** Read as 00000_B after reset; returns value that was written.
PC6	[23:19]	rw	**Port Control for Port 11 Pin 6** (coding see **Table 14-5**)
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

14.7.1.13 Port 11 Input/Output Control Register 12

The PC13, PC14 and PC15 bit fields in register P00_IOCR12 are not connected.

P11_IOCR12
Port 11 Input/Output Control Register 12

($1C_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
PC13, PC14, PC15	[15:11], [23:19], [31:27]	rw	**Reserved** Read as 00000_B after reset; returns value that was written.
PC12	[7:3]	rw	**Port Control for Port 11.12** (coding see **Table 14-5**)
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

14.7.1.14 Port 11 Input Register

The basic P11_IN register functionality is described on **Page 14-40**. However, port lines P11.[15:13] are not connected. Therefore, bits P[15:9] in register P10_IN are always read as 0.

14.7.1.15 Port 11 Pad Driver Mode 0 Register

The basic P11_PDR0 register functionality is described on **Page 14-20**.

P11_PDR0
Port 11 Pad Driver Mode 0 Register (40_H) **Reset Value: $3333\ 3333_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PL7	**PD7**			**PL6**	**PD6**			**PL5**	**PD5**			**PL4**	**PD4**		
rw	rw			rw	rw			rw	rw			rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PL3	**PD3**			**PL2**	**PD2**			**PL1**	**PD1**			**PL0**	**PD0**		
rw	rw			rw	rw			rw	rw			rw	rw		

Field	Bits	Type	Description
PD2	[10:8]	rw	**Pad Driver Mode for Port 11 Pin 2**
PD3	[14:12]	rw	**Pad Driver Mode for Port 11 Pin 3**
PD6	[26:24]	rw	**Pad Driver Mode for Port 11 Pin 6**
PD7	[30:28]	rw	**Pad Driver Mode for Port 11 Pin 7**
PL2	11	rw	**Pad Level Selection for Port 11 Pin 2**
PL3	15	rw	**Pad Level Selection for Port 11 Pin 3**
PL6	27	rw	**Pad Level Selection for Port 11 Pin 6**
PL7	31	rw	**Pad Level Selection for Port 11 Pin 7**
PD0, PD1, PD4, PD5	[2:0], [6:4], [18:16], [22:20]	rw	**Reserved** Read as 000_B after reset; returns value that was written.
PL0, PL1, PL4, PL5	3, 7, 19, 23	rw	**Reserved** Read as 0_B after reset; returns value that was written.

14.7.1.16 Port 11 Pad Driver Mode 1 Register

P11_PDR1
Port 11 Pad Driver Mode 1 Register (44$_H$) **Reset Value: 3333 3333$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PL15		PD15		PL14		PD14		PL13		PD13		PL12		PD12	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PL11		PD11		PL10		PD10		PL9		PD9		PL8		PD8	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
PD8	[2:0]	rw	**Pad Driver Mode for Port 11 Pin 8**
PD9	[6:4]	rw	**Pad Driver Mode for Port 11 Pin 9**
PD10	[10:8]	rw	**Pad Driver Mode for Port 11 Pin 10**
PD11	[14:12]	rw	**Pad Driver Mode for Port 11 Pin 11**
PD12	[18:16]	rw	**Pad Driver Mode for Port 11 Pin 12**
PL9	7	rw	**Pad Level Selection for Port 11 Pin 9**
PL10	11	rw	**Pad Level Selection for Port 11 Pin 10**
PL11	15	rw	**Pad Level Selection for Port 11 Pin 11**
PL12	19	rw	**Pad Level Selection for Port 11 Pin 12**
PD13, PD14, PD15	[22:20], [26:24], [30:28]	rw	**Reserved** Read as 011_B after reset; returns value that was written, must be written with reset value.
PL8, PL13, PL14, PL15	3, 23, 27, 31	rw	**Reserved** Read as 0, returns values last written, must be written with 0.

14.7.1.17 Port 11 Emergency Stop Register

The basic P11_ESR register functionality is described on **Page 14-38**. Port lines P11.[15:13], P11.7, P11.[5:4], P11.[1:0] are not connected. Reading the P10_ESR bits EN[15:13], EN7, EN[5:4], EN[1:0] returns value that was last written..

14.8 Port 13

This section describes the Port 13 functionality.

Port 13 is an 4-bit GPIO port.

14.8.1 Port 13 Registers

The following registers are available on Port 13:

Table 14-12 Port 13 Registers

Register Short Name	Register Long Name	Offset[1] Address	Description see
P13_OUT	Port 13 Output Register	0000_H	Page 14-70[2]
P13_OMR	Port 13 Output Modification Register	0004_H	Page 14-70[2]
P13_IOCR0	Port 13 Input/Output Control Register 0	0010_H	Page 14-14
P13_IN	Port 13 Input Register	0024_H	Page 14-71[2]
P13_PDR0	Port 13 Pad Driver Mode 0 Register	0040_H	Page 14-72[2]
P13_ESR	Port 13 Emergency Stop Register	0050_H	Page 14-72[2]
P13_OMSR0	Port 13 Output Modification Set Register 0	0070_H	Page 14-29
P13_OMCR0	Port 13 Output Modification Clear Register 0	0080_H	Page 14-34
P13_OMSR	Port 13 Output Modification Set Register	0090_H	Page 14-71[2]
P13_OMCR	Port 13 Output Modification Clear Register	0094_H	Page 14-71[2]
P13_ACCEN1	Port 13 Access Enable Register 1	$00F8_H$	Page 14-43
P13_ACCEN0	Port 13 Access Enable Register 0	$00FC_H$	Page 14-41

1) The absolute addresses are calculated by adding the offset address to the module base address (see Table 14-3)

2) These registers are listed and noted here in the Port 13 section because they differ from the general port register description given in Section 14.3.

14.8.1.1 Port 13 Output Register

The basic P13_OUT register functionality is described on Page 14-24. Port line P13.[15:4] are not connected. Reading the P13_OUT bit P[15:4] always return 0. These connected bits can be also set/reset by the corresponding bits in P13_OMR, P13_OMSR, P13_OMCR, P13_OMSR0, P13_OMCR0.

14.8.1.2 Port 13 Output Modification Register

The basic P13_OMR register functionality is described on Page 14-26. However, port lines P13.[15:4] are not connected.

14.8.1.3 Port 13 Output Modification Set Register

The basic P13_OMSR register functionality is described on **Page 14-28**. However, port lines P13.[15:4] are not connected.

14.8.1.4 Port 13 Output Modification Clear Register

The basic P13_OMCR register functionality is described on **Page 14-33**. However, port lines P13.[15:4] are not connected.

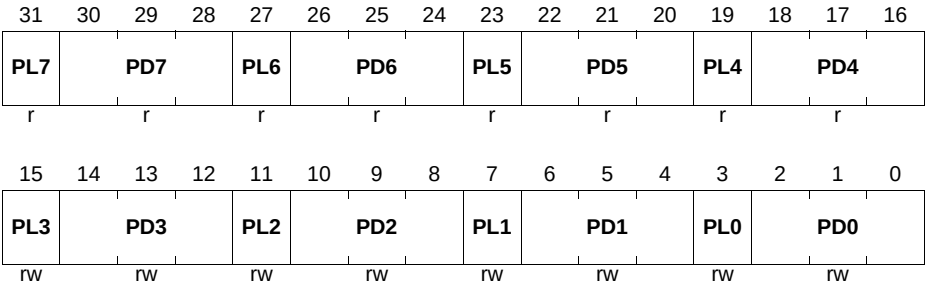
14.8.1.5 Port 13 Input Register

The basic P13_IN register functionality is described on **Page 14-40**. However, port lines P13.[15:4] are not connected. Therefore, bits P[15:4] in register P13_IN are always read as 0.

14.8.1.6 Port 13 Pad Driver Mode 0 Register

The basic P13_PDR0 register functionality is described on **Page 14-20**.

P13_PDR0
Port 13 Pad Driver Mode 0 Register (40_H) **Reset Value: $0000\ 3333_H$**



Field	Bits	Type	Description
PD0, **PD1,** **PD2,** **PD3**	[2:0], [6:4], [10:8], [14:12]	rw	**Pad Driver Mode for Port 13 Pin 0 to 3**
PL0, **PL1,** **PL2,** **PL3**	3, 7, 11, 15	rw	**Reserved** Read as 0, returns values last written, must be written with 0.
PD4, **PD5,** **PD6,** **PD7**	[18:16], [22:20], [26:24], [30:28]	r	**Reserved** Read as 000_B after reset; should be written with 0.
PL4, **PL5,** **PL6,** **PL7**	19, 23, 27, 31	r	**Reserved** Read as 0_B after reset; should be written with 0.

14.8.1.7 Port 13 Emergency Stop Register

The basic P13_ESR register functionality is described on **Page 14-38**. Port lines P13.[15:4] are not connected. Reading the P13_ESR bits EN[15:4] always returns 0.

14.9 Port 14

This section describes the Port 14 functionality.

14.9.1 Port 14 Registers

The following registers are available on Port 14:

Table 14-13 Port 14 Registers

Register Short Name	Register Long Name	Offset[1] Address	Description see
P14_OUT	Port 14 Output Register	0000_H	Page 14-51 [2]
P14_OMR	Port 14 Output Modification Register	0004_H	Page 14-51 [2]
P14_IOCR0	Port 14 Input/Output Control Register 0	0010_H	Page 14-14
P14_IOCR4	Port 14 Input/Output Control Register 4	0014_H	Page 14-15
P14_IOCR8	Port 14 Input/Output Control Register 8	0018_H	Page 14-52 [2]
P14_IN	Port 14 Input Register	0024_H	Page 14-52 [2]
P14_PDR0	Port 14 Pad Driver Mode 0 Register	0040_H	Page 14-20
P14_PDR1	Port 14 Pad Driver Mode 1 Register	0044_H	Page 14-53 [2]
P14_ESR	Port 14 Emergency Stop Register	0050_H	Page 14-53 [2]
P14_OMSR0	Port 14 Output Modification Set Register 0	0070_H	Page 14-29
P14_OMSR4	Port 14 Output Modification Set Register 4	0074_H	Page 14-30
P14_OMSR8	Port 14 Output Modification Set Register 8	0078_H	Page 14-51 [2]
P14_OMCR0	Port 14 Output Modification Clear Register 0	0080_H	Page 14-34
P14_OMCR4	Port 14 Output Modification Clear Register 4	0084_H	Page 14-35
P14_OMCR8	Port 14 Output Modification Clear Register 8	0088_H	Page 14-51 [2]
P14_OMSR	Port 14 Output Modification Set Register	0090_H	Page 14-51 [2]
P14_OMCR	Port 14 Output Modification Clear Register	0094_H	Page 14-51 [2]
P14_ACCEN1	Port 14 Access Enable Register 1	$00F8_H$	Page 14-43
P14_ACCEN0	Port 14 Access Enable Register 0	$00FC_H$	Page 14-41

1) The absolute addresses are calculated by adding the offset address to the module base address (see **Table 14-3**)
2) These registers are listed and noted here in the Port 14 section because they differ from the general port register description given in **Section 14.3**.

14.9.1.1 Port 14 Output Register

The basic P14_OUT register functionality is described on **Page 14-24**. Port lines P14.[15:9] are not connected. Reading the P14_OUT bits P[15:12] always return 0. Reading P[11:9] returns the value that was last written. These connected bits can be also set/reset by the corresponding bits in P14_OMR, P14_OMSR, P14_OMCR, P14_OMSRx (x=0,4,8), P14_OMCRx (x=0,4,8).

14.9.1.2 Port 14 Output Modification Register

The basic P14_OMR register functionality is described on **Page 14-26**. However, port lines P14.[15:9] are not connected. The P14_OMR bits PS[11:9] and PCL[11:9] have no direct effect on port lines but only on register bits P14_OUT.P[11:9].

14.9.1.3 Port 14 Output Modification Set Register

The basic P14_OMSR register functionality is described on **Page 14-28**. However, port lines P14.[15:9] are not connected. The P14_OMSR bits PS[11:9] have no direct effect on port lines but only on register bits P14_OUT.P[11:9].

14.9.1.4 Port 14 Output Modification Set Register 8

The basic P14_OMSR8 register functionality is described on **Page 14-31**. However, port lines P14.[11:9] are not connected. The P14_OMSR8 bits PS[11:9] have no direct effect on port lines but only on register bits P14_OUT.P[11:9].

14.9.1.5 Port 14 Output Modification Clear Register

The basic P14_OMCR register functionality is described on **Page 14-33**. However, port lines P14.[15:9] are not connected. The P14_OMCR bits PCL[11:9] have no direct effect on port lines but only on register bits P14_OUT.P[11:9].

14.9.1.6 Port 14 Output Modification Clear Register 8

The basic P14_OMCR8 register functionality is described on **Page 14-36**. However, port lines P14.[11:9] are not connected. The P14_OMCR8 bits PCL[11:9] have no direct effect on port lines but only on register bits P14_OUT.P[11:9].

14.9.1.7 Port 14 Input/Output Control Register 8

The PC[11:9] bit fields in register P14_IOCR8 are not connected.

P14_IOCR8
Port 14 Input/Output Control Register 8

(18_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PC11					0			PC10					0		
rw					r			rw					r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC9					0			PC8					0		
rw					r			rw					r		

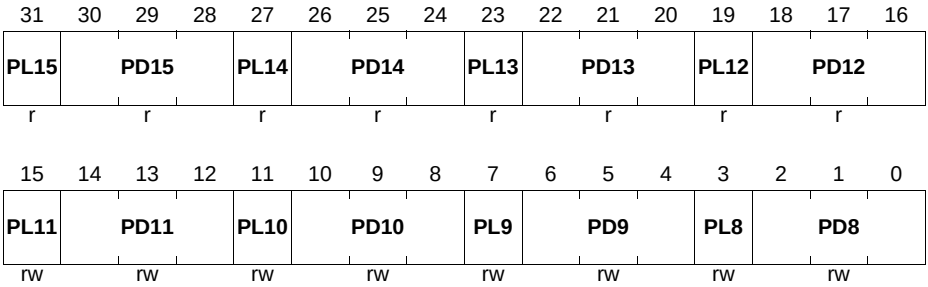
Field	Bits	Type	Description
PC8	[7:3]	rw	**Port Control for Port 14 Pin 8** Coding see **Table 14-5**.
PC9, PC10, PC11	[15:11], [23:19], [31:27]	rw	**Reserved** Read as 00000_B after reset; returns value that was written.
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

14.9.1.8 Port 14 Input Register

The basic P14_IN register functionality is described on **Page 14-40**. However, port lines P14.[15:9] are not connected. Therefore, bits P[15:9] in register P14_IN are always read as 0.

14.9.1.9 P14 Pad Driver Mode 1 Register

P14_PDR1
Port 14 Pad Driver Mode 1 Register (44_H) **Reset Value: $0000\ 3333_H$**



Field	Bits	Type	Description
PD8	[2:0]	rw	**Pad Driver Mode for Port n Pin 8**
PD9, PD10, PD11	[6:4], [10:8], [14:12]	rw	**Reserved** Read as 011_B after reset; returns value that was written, must be written with reset value.
PD12, PD13, PD14, PD15	[18:16], [22:20], [26:24], [30:28]	r	**Reserved** Read as 000_B after reset; should be written as 0.
PL8, PL9, PL10, PL11	3, 7, 11, 15	rw	**Reserved** Read as 0_B after reset; returns value that was written, have to be written with 0.
PL12, PL13, PL14, PL15	19, 23, 27, 31	r	**Reserved** Read as 0_B after reset; should be written as 0.

14.9.1.10 Port 14 Emergency Stop Register

The basic P14_ESR register functionality is described on **Page 14-38**. Port lines P14.[15:9] are not connected. Reading the P14_ESR bits EN[15:12] always returns 0. Reading the P14_ESR bits EN[11:9] returns value that was last written.

14.10 Port 15

This section describes the Port 15 functionality.

14.10.1 Port 15 Registers

The following registers are available on Port 15:

Table 14-14 Port 14 Registers

Register Short Name	Register Long Name	Offset[1] Address	Description see
P15_OUT	Port 15 Output Register	0000_H	Page 14-51 [2]
P15_OMR	Port 15 Output Modification Register	0004_H	Page 14-51 [2]
P15_IOCR0	Port 15 Input/Output Control Register 0	0010_H	Page 14-14
P15_IOCR4	Port 15 Input/Output Control Register 4	0014_H	Page 14-15
P15_IOCR8	Port 15 Input/Output Control Register 8	0018_H	Page 14-52 [2]
P15_IN	Port 15 Input Register	0024_H	Page 14-52 [2]
P15_PDR0	Port 15 Pad Driver Mode 0 Register	0040_H	Page 14-20
P15_PDR1	Port 15 Pad Driver Mode 1 Register	0044_H	Page 14-53 [2]
P15_ESR	Port 15 Emergency Stop Register	0050_H	Page 14-53 [2]
P15_OMSR0	Port 15 Output Modification Set Register 0	0070_H	Page 14-29
P15_OMSR4	Port 15 Output Modification Set Register 4	0074_H	Page 14-30
P15_OMSR8	Port 15 Output Modification Set Register 8	0078_H	Page 14-51 [2]
P15_OMCR0	Port 15 Output Modification Clear Register 0	0080_H	Page 14-34
P15_OMCR4	Port 15 Output Modification Clear Register 4	0084_H	Page 14-35
P15_OMCR8	Port 15 Output Modification Clear Register 8	0088_H	Page 14-51 [2]
P15_OMSR	Port 15 Output Modification Set Register	0090_H	Page 14-51 [2]
P15_OMCR	Port 15 Output Modification Clear Register	0094_H	Page 14-51 [2]
P15_ACCEN1	Port 15 Access Enable Register 1	$00F8_H$	Page 14-43
P15_ACCEN0	Port 15 Access Enable Register 0	$00FC_H$	Page 14-41

1) The absolute addresses are calculated by adding the offset address to the module base address (see **Table 14-3**)
2) These registers are listed and noted here in the Port 14 section because they differ from the general port register description given in **Section 14.3**.

14.10.1.1 Port 15 Output Register

The basic P15_OUT register functionality is described on **Page 14-24**. Port lines P15.[15:9] are not connected. Reading the P15_OUT bits P[15:12] always return 0. Reading P[11:9] returns the value that was last written. These connected bits can be also set/reset by the corresponding bits in P15_OMR, P15_OMSR, P15_OMCR, P15_OMSRx (x=0,4,8), P15_OMCRx (x=0,4,8).

14.10.1.2 Port 15 Output Modification Register

The basic P15_OMR register functionality is described on **Page 14-26**. However, port lines P15.[15:9] are not connected. The P15_OMR bits PS[11:9] and PCL[11:9] have no direct effect on port lines but only on register bits P15_OUT.P[11:9].

14.10.1.3 Port 15 Output Modification Set Register

The basic P15_OMSR register functionality is described on **Page 14-28**. However, port lines P15.[15:9] are not connected. The P15_OMSR bits PS[11:9] have no direct effect on port lines but only on register bits P15_OUT.P[11:9].

14.10.1.4 Port 15 Output Modification Set Register 8

The basic P15_OMSR8 register functionality is described on **Page 14-31**. However, port lines P15.[11:9] are not connected. The P15_OMSR8 bits PS[11:9] have no direct effect on port lines but only on register bits P15_OUT.P[11:9].

14.10.1.5 Port 15 Output Modification Clear Register

The basic P15_OMCR register functionality is described on **Page 14-33**. However, port lines P15.[15:9] are not connected. The P15_OMCR bits PCL[11:9] have no direct effect on port lines but only on register bits P15_OUT.P[11:9].

14.10.1.6 Port 14 Output Modification Clear Register 8

The basic P15_OMCR8 register functionality is described on **Page 14-36**. However, port lines P15.[11:9] are not connected. The P15_OMCR8 bits PCL[11:9] have no direct effect on port lines but only on register bits P15_OUT.P[11:9].

14.10.1.7 Port 14 Input/Output Control Register 8

The PC[11:9] bit fields in register P15_IOCR8 are not connected.

P15_IOCR8
Port 15 Input/Output Control Register 8

(18_H) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PC11					0			PC10					0		
rw					r			rw					r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC9					0			PC8					0		
rw					r			rw					r		

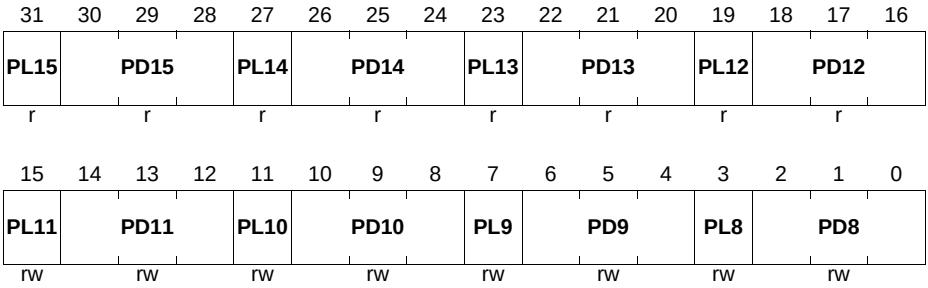
Field	Bits	Type	Description
PC8	[7:3]	rw	**Port Control for Port 15 Pin 8** Coding see **Table 14-5**.
PC9, PC10, PC11	[15:11], [23:19], [31:27]	rw	**Reserved** Read as 00000_B after reset; returns value that was written.
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

14.10.1.8 Port 15 Input Register

The basic P15_IN register functionality is described on **Page 14-40**. However, port lines P15.[15:9] are not connected. Therefore, bits P[15:9] in register P15_IN are always read as 0.

14.10.1.9 P15 Pad Driver Mode 1 Register

P15_PDR1
Port 15 Pad Driver Mode 1 Register (44$_H$) **Reset Value: 0000 3333$_H$**



Field	Bits	Type	Description
PD8	[2:0]	rw	**Pad Driver Mode for Port n Pin 8**
PD9, PD10, PD11	[6:4], [10:8], [14:12]	rw	**Reserved** Read as 011_B after reset; returns value that was written, must be written with reset value.
PD12, PD13, PD14, PD15	[18:16], [22:20], [26:24], [30:28]	r	**Reserved** Read as 000_B after reset; should be written as 0.
PL8, PL9, PL10, PL11	3, 7, 11, 15	rw	**Reserved** Read as 0_B after reset; returns value that was written, have to be written with 0.
PL12, PL13, PL14, PL15	19, 23, 27, 31	r	**Reserved** Read as 0_B after reset; should be written as 0.

14.10.1.10Port 15 Emergency Stop Register

The basic P15_ESR register functionality is described on **Page 14-38**. Port lines P15.[15:9] are not connected. Reading the P15_ESR bits EN[15:12] always returns 0. Reading the P15_ESR bits EN[11:9] returns value that was last written.

14.11 Port 20

This section describes the Port 20 functionality.

14.11.1 Port 20 Registers

The following registers are available on Port 20:

Table 14-15 Port 20 Registers

Register Short Name	Register Long Name	Offset[1] Address	Description see
P20_OUT	Port 20 Output Register	0000_H	Page 14-85[2]
P20_OMR	Port 20 Output Modification Register	0004_H	Page 14-85[2]
P20_IOCR0	Port 20 Input/Output Control Register 0	0010_H	Page 14-87[2]
P20_IOCR4	Port 20 Input/Output Control Register 4	0014_H	Page 14-88[2]
P20_IOCR8	Port 20 Input/Output Control Register 8	0018_H	Page 14-16
P20_IOCR12	Port 20 Input/Output Control Register 12	$001C_H$	Page 14-89[2]
P20_IN	Port 20 Input Register	0024_H	Page 14-89[2]
P20_PDR0	Port 20 Pad Driver Mode 0 Register	0040_H	Page 14-90[2]
P20_PDR1	Port 20 Pad Driver Mode 1 Register	0044_H	Page 14-91[2]
P20_ESR	Port 20 Emergency Stop Register	0050_H	Page 14-92[2]
P20_OMSR0	Port 20 Output Modification Set Register 0	0070_H	Page 14-85[2]
P20_OMSR4	Port 20 Output Modification Set Register 4	0074_H	Page 14-85[2]
P20_OMSR8	Port 20 Output Modification Set Register 8	0078_H	Page 14-31
P20_OMSR12	Port 20 Output Modification Set Register 12	$007C_H$	Page 14-85[2]
P20_OMCR0	Port 20 Output Modification Clear Register 0	0080_H	Page 14-86[2]
P20_OMCR4	Port 20 Output Modification Clear Register 4	0084_H	Page 14-86[2]
P20_OMCR8	Port 20 Output Modification Clear Register 8	0088_H	Page 14-36
P20_OMCR12	Port 20 Output Modification Clear Register 12	$008C_H$	Page 14-86[2]
P20_OMSR	Port 20 Output Modification Set Register	0090_H	Page 14-85[2]
P20_OMCR	Port 20 Output Modification Clear Register	0094_H	Page 14-86[2]
P20_ACCEN1	Port 20 Access Enable Register 1	$00F8_H$	Page 14-43
P20_ACCEN0	Port 20 Access Enable Register 0	$00FC_H$	Page 14-41

1) The absolute addresses are calculated by adding the offset address to the module base address (see Table 14-3)

2) These registers are listed and noted here in the Port 20 section because they differ from the general port register description given in Section 14.3.

14.11.1.1 Port 20 Output Register

The basic P20_OUT register functionality is described on **Page 14-24**. Port line P20.1, P20.[5:4] and P20.15 are not connected. Reading the P20_OUT bits P1, P[5:4] and P15 returns the value that was last written (0 after reset). These connected bits can be also set/reset by the corresponding bits in P20_OMR, P20_OMSR, P20_OMCR, P20_OMSRx (x=0, 4,12), P20_OMCRx (x=0, 4,12). P20_OUT.P2 does not determine the level at P20.2, returns the value that was last written.

14.11.1.2 Port 20 Output Modification Register

The basic P20_OMR register functionality is described on **Page 14-26**. However, port lines P20.1, P20.[5:4] and P20.15 are not connected. The P20_OMR bits PS1, PS[5:4], PS15 and PCL1, PCL[5:4], PCL15 have no direct effect on port lines but only on register bits P20_OUT P1, P[5:4] and P15. P20_OMR bit PS2 and PCL2 has no effect on P20_OUT.P2.

14.11.1.3 Port 20 Output Modification Set Register

The basic P20_OMSR register functionality is described on **Page 14-28**. However, port lines P20.1, P20.[5:4], P20.15 is not connected. The P20_OMSR bits PS1, PS.[5:4], PS15 has no direct effect on port line but only on register bits P20_OUT.P1, P[5:4], P15. P20_OMSR bit PS2 has no effect on P20_OUT.P2.

14.11.1.4 Port 20 Output Modification Set Register 0

The basic P20_OMSR0 register functionality is described on **Page 14-29**. However, port lines P20.1 are not connected. The P20_OMSR0 bits PS1 has no direct effect on port lines but only on register bits P20_OUT.P1. P20_OMSR0 bit PS2 has no effect on P20_OUT.P2.

14.11.1.5 Port 20 Output Modification Set Register 4

The basic P20_OMSR4 register functionality is described on **Page 14-30**. However, port lines P20.[5:4] are not connected. The P20_OMSR4 bits PS[5:4] has no direct effect on port lines but only on register bits P20_OUT.P[5:4].

14.11.1.6 Port 20 Output Modification Set Register 12

The basic P20_OMSR12 register functionality is described on **Page 14-32**. However, port lines P20.15 is not connected. The P20_OMSR12 bits PS15 has no direct effect on port lines but only on register bits P20_OUT.P15.

14.11.1.7 Port 20 Output Modification Clear Register

The basic P20_OMCR register functionality is described on **Page 14-33**. However, port lines P20.1, P20.[5:4] and P20.15 are not connected. The P20_OMCR bits PCL1, PCL[5:4], PCL15 has no direct effect on port lines but only on register bits P20_OUT.P1, P[5:4] and P15. P20_OMCR bit PCL2 has no effect on P20_OUT.P2.

14.11.1.8 Port 20 Output Modification Clear Register 0

The basic P20_OMCR0 register functionality is described on **Page 14-34**. However, port lines P20.1 are not connected. The P20_OMCR0 bits PCL1 has no direct effect on port lines but only on register bits P20_OUT.P1. P20_OMCR0 bit PCL2 has no effect on P20_OUT.P2.

14.11.1.9 Port 20 Output Modification Clear Register 4

The basic P20_OMCR4 register functionality is described on **Page 14-35**. However, port lines P20.[5:4] are not connected. The P20_OMCR4 bits PCL[5:4] has no direct effect on port lines but only on register bits P20_OUT.P[5:4].

14.11.1.10Port 20 Output Modification Clear Register 12

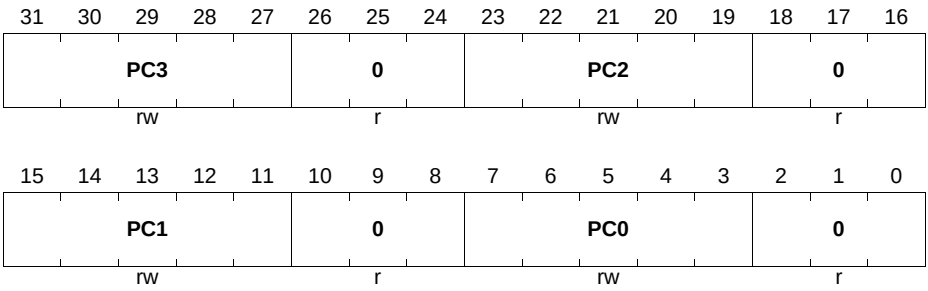
The basic P20_OMCR12 register functionality is described on **Page 14-37**. However, port lines P20.15 are not connected. The P20_OMCR12 bits PCL15 has no direct effect on port lines but only on register bits P20_OUT.P15.

14.11.1.11Port 20 Input/Output Control Register 0

The PC1 bit field in register P20_IOCR0 are not connected.

P20_IOCR0
Port 20 Input/Output Control Register 0

(10_H) **Reset Value: $0010\ 0000_H$**



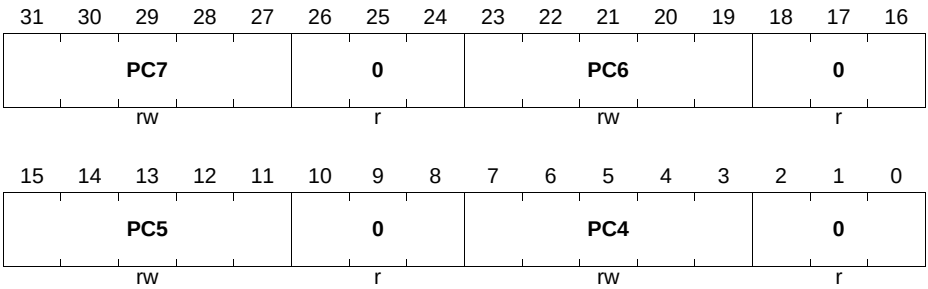
Field	Bits	Type	Description
PC1	[15:11]	rw	**Reserved** Read as 00000_B after reset; returns value that was written.
PC0	[7:3]	rw	**Port Control for Port 20 Pin 0** (coding see **Table 14-5**)
PC2	[23:19]	rw	**Reserved** Read as 00010_B after reset; returns value that was written.
PC3	[31:27]	rw	**Port Control for Port 20 Pin 3** (coding see **Table 14-5**)
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

14.11.1.12Port 20 Input/Output Control Register 4

The PC[5:4] bit fields in register P20_IOCR4 are not connected.

P20_IOCR4
Port 20 Input/Output Control Register 4

(14_H) **Reset Value: 0000 0000$_H$**



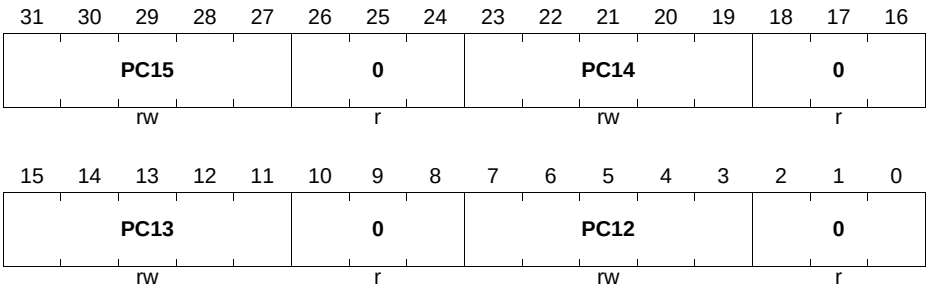
Field	Bits	Type	Description
PC4, PC5	[7:3], [15:11]	rw	**Reserved** Read as 00000_B after reset; returns value that was written.
PC6	[23:19]	rw	**Port Control for Port 20 Pin 6** (coding see **Table 14-5**)
PC7	[31:27]	rw	**Port Control for Port 20 Pin 7** (coding see **Table 14-5**)
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

14.11.1.13Port 20 Input/Output Control Register 12

The PC15 bit fields in register P20_IOCR12 is not connected.

P20_IOCR12
Port 20 Input/Output Control Register 12

($1C_H$) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
PC15	[31:27]	rw	**Reserved** Read as 00000_B after reset; returns value that was written.
PC12	[7:3]	rw	**Port Control for Port 20 Pin 12** (coding see **Table 14-5**)
PC13	[15:11]	rw	**Port Control for Port 20 Pin 13** (coding see **Table 14-5**)
PC14	[23:19]	rw	**Port Control for Port 20 Pin 14** (coding see **Table 14-5**)
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

14.11.1.14Port 20 Input Register

The basic P20_IN register functionality is described on **Page 14-40**. However, port lines P20.1, P20.[5:4] and P20.15 are not connected. Therefore, bits P1, P[5:4] and P15 in register P20_IN are always read as 0.

14.11.1.15Port 20 Pad Driver Mode 0 Register

The basic P20_PDR0 register functionality is described on **Page 14-20**.

P20_PDR0
Port 20 Pad Driver Mode 0 Register (40_H) **Reset Value: $3333\ 3333_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PL7	**PD7**			**PL6**	**PD6**			**PL5**	**PD5**			**PL4**	**PD4**		
rw	rw			rw	rw			rw	rw			rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PL3	**PD3**			**PL2**	**PD2**			**PL1**	**PD1**			**PL0**	**PD0**		
rw	rw			rw	rw			rw	rw			rw	rw		

Field	Bits	Type	Description
PD0	[2:0]	rw	**Pad Driver Mode for Port 20 Pin 0**
PD3	[14:12]	rw	**Pad Driver Mode for Port 20 Pin 3**
PD6	[26:24]	rw	**Pad Driver Mode for Port 20 Pin 6**
PD7	[30:28]	rw	**Pad Driver Mode for Port 20 Pin 7**
PD1, **PD2,** **PD4,** **PD5**	[6:4], [10:8], [18:16], [22:20]	rw	**Reserved** Read as 011_B after reset; returns value that was written, must be written with reset value.
PL0, **PL1,** **PL2,** **PL3,** **PL4,** **PL5,** **PL6,** **PL7**	3, 7, 11, 15, 19, 23, 27, 31	rw	**Reserved** Read as 0, returns values last written, must be written with 0.

14.11.1.16Port 20 Pad Driver Mode 1 Register

P20_PDR1
Port 20 Pad Driver Mode 1 Register (44$_H$) **Reset Value: 3333 3333$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PL15		PD15		PL14		PD14		PL13		PD13		PL12		PD12	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PL11		PD11		PL10		PD10		PL9		PD9		PL8		PD8	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
PD8	[2:0]	rw	**Pad Driver Mode for Port 20 Pin 8**
PD9	[6:4]	rw	**Pad Driver Mode for Port 20 Pin 9**
PD10	[10:8]	rw	**Pad Driver Mode for Port 20 Pin 10**
PD11	[14:12]	rw	**Pad Driver Mode for Port 20 Pin 11**
PD12	[18:16]	rw	**Pad Driver Mode for Port 20 Pin 12**
PD13	[22:20]	rw	**Pad Driver Mode for Port 20 Pin 13**
PD14	[26:24]	rw	**Pad Driver Mode for Port 20 Pin 14**
PD15	[30:28]	rw	**Reserved** Read as 011$_B$ after reset; returns value that was written, must be written with reset value.
PL8, **PL9,** **PL10,** **PL11,** **PL12,** **PL13,** **PL14,** **PL15**	3, 7, 11, 15, 19, 23, 27, 31	rw	**Reserved** Read as 0, returns values last written, must be written with 0.

14.11.1.17Port 20 Emergency Stop Register

The basic P20_ESR register functionality is described on **Page 14-38**. Port lines P20.1, P20.[5:4] and P20.15 are not connected. Reading the EN1, EN2, EN[5:4], EN15 returns value that was last written.

14.12 Port 21

This section describes the Port 21 functionality.

Port 21 is an 8-bit GPIO port.

14.12.1 Port 21 Registers

The following registers are available on Port 21:

Table 14-16 Port 21 Registers

Register Short Name	Register Long Name	Offset[1] Address	Description see
P21_OUT	Port 21 Output Register	0000_H	Page 14-94[2]
P21_OMR	Port 21 Output Modification Register	0004_H	Page 14-95[2]
P21_IOCR0	Port 21 Input/Output Control Register 0	0010_H	Page 14-96[2]
P21_IOCR4	Port 21 Input/Output Control Register 4	0014_H	Page 14-15
P21_IN	Port 21 Input Register	0024_H	Page 14-98[2]
P21_PDR0	Port 21 Pad Driver Mode 0 Register	0040_H	Page 14-97[2]
P21_ESR	Port 21 Emergency Stop Register	0050_H	Page 14-98[2]
P21_OMSR0	Port 21 Output Modification Set Register 0	0070_H	Page 14-95[3]
P21_OMSR4	Port 21 Output Modification Set Register 4	0074_H	Page 14-30
P21_OMCR0	Port 21 Output Modification Clear Register 0	0080_H	Page 14-95[4]
P21_OMCR4	Port 21 Output Modification Clear Register 4	0084_H	Page 14-35
P21_OMSR	Port 21 Output Modification Set Register	0090_H	Page 14-95[5]
P21_OMCR	Port 21 Output Modification Clear Register	0094_H	Page 14-95[6]
P21_ACCEN1	Port 21 Access Enable Register 1	$00F8_H$	Page 14-43
P21_ACCEN0	Port 21 Access Enable Register 0	$00FC_H$	Page 14-41

1) The absolute addresses are calculated by adding the offset address to the module base address (see Table 14-3)

2) These registers are listed and noted here in the Port 21 section because they differ from the general port register description given in Section 14.3.

3) These registers are listed and noted here in the Port 21 section because they differ from the general port register description given in Section 14.3.

4) These registers are listed and noted here in the Port 21 section because they differ from the general port register description given in Section 14.3.

5) These registers are listed and noted here in the Port 21 section because they differ from the general port register description given in Section 14.3.

6) These registers are listed and noted here in the Port 21 section because they differ from the general port register description given in Section 14.3.

14.12.1.1 Port 21 Output Register

The basic P21_OUT register functionality is described on Page 14-24. Port line P21.[1:0], [15:8] are not connected. Reading the P21_OUT bit P[15:8] always return 0,

reading P21_OUT bit P[1:0] returns the value that was last written. These connected bits can be also set/reset by the corresponding bits in P21_OMR, P21_OMSR, P21_OMCR, P21_OMSRx (x=0,4), P21_OMCRx (x=0,4).

14.12.1.2 Port 21 Output Modification Register

The basic P21_OMR register functionality is described on **Page 14-26**. However, port lines P21.[1:0], P21.[15:8] are not connected. The P21_OMR bits PS[1:0] and PCL[1:0] have no direct effect on port lines but only on register bits P21_OUT.P[1:0].

14.12.1.3 Port 21 Output Modification Set Register

The basic P21_OMSR register functionality is described on **Page 14-28**. However, port lines P21.[1:0], P21.[15:8] is not connected. The P21_OMR bits PS[1:0] have no direct effect on port lines but only on register bits P21_OUT.P[1:0].

14.12.1.4 Port 21 Output Modification Set Register 0

The basic P21_OMSR0 register functionality is described on **Page 14-29**. However, port lines P21.[1:0] are not connected. The P21_OMSR0 bits PS[1:0] have no direct effect on port lines but only on register bits P21_OUT.P[1:0].

14.12.1.5 Port 21 Output Modification Clear Register

The basic P21_OMCR register functionality is described on **Page 14-33**. However, port lines P21.[15:8] are not connected. The P21_OMCR bits PCL[1:0] have no direct effect on port lines but only on register bits P21_OUT.P[1:0].

14.12.1.6 Port 21 Output Modification Clear Register 0

The basic P21_OMCR0 register functionality is described on **Page 14-29**. However, port lines P21.[1:0] are not connected. The P21_OMCR0 bits PCL[1:0] have no direct effect on port lines but only on register bits P21_OUT.P[1:0].

14.12.1.7 Port 21 Input/Output Control Register 0

The PC[1:0] bit fields in register P21_IOCR0 are not connected.

P21_IOCR0
Port 21 Input/Output Control Register 0

(10_H) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PC3					**0**			**PC2**					**0**		
rw					r			rw					r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC1					**0**			**PC0**					**0**		
rw					r			rw					r		

Field	Bits	Type	Description
PC2	[23:19]	rw	**Port Control for Port 21 Pin 2**
PC3	[31:27]	rw	**Port Control for Port 21 Pin 3**
PC0, PC1	[7:3], [15:11]	rw	**Reserved** Read as 00000_B after reset; returns value that was written.
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

14.12.1.8 Port 21 Pad Driver Mode 0 Register

The basic P21_PDR0 register functionality is described on **Page 14-20**.

P21_PDR0
Port 21 Pad Driver Mode 0 Register (40$_H$) **Reset Value: 3333 3333$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PL7	**PD7**			**PL6**	**PD6**			**PL5**	**PD5**			**PL4**	**PD4**		
rw	rw			rw	rw			rw	rw			rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PL3	**PD3**			**PL2**	**PD2**			**PL1**	**PD1**			**PL0**	**PD0**		
rw	rw			rw	rw			rw	rw			rw	rw		

Field	Bits	Type	Description
PD2	[10:8]	rw	**Pad Driver Mode for Port 21 Pin 2**
PD3	[14:12]	rw	**Pad Driver Mode for Port 21 Pin 3**
PD4	[18:16]	rw	**Pad Driver Mode for Port 21 Pin 4**
PD5	[22:20]	rw	**Pad Driver Mode for Port 21 Pin 5**
PD6	[26:24]	rw	**Pad Driver Mode for Port 21 Pin 6**
PD7	[30:28]	rw	**Pad Driver Mode for Port 21 Pin 7**
PD0, **PD1**	[2:0], [6:4]	rw	**Reserved** Read as 011_B after reset; returns value that was written, must be written with reset value.
PL0, **PL1,** **PL2,** **PL3,** **PL4,** **PL5,** **PL6,** **PL7**	3, 7, 11, 15, 19, 23, 27, 31	rw	**Reserved** Read as 0, returns values last written, must be written with 0.

14.12.1.9 Port 21 Input Register

The basic P21_IN register functionality is described on **Page 14-40**. However, port lines P21.[1:0], P21.[15:8] are not connected. Therefore, bits P[1:0], P[15:8] in register P21_IN are always read as 0.

14.12.1.10 Port 21 Emergency Stop Register

The basic P21_ESR register functionality is described on **Page 14-38**. Port lines P21.[1:0], P21.[15:8] are not connected. Reading the P15_ESR bits EN[15:8] always returns 0, reading EN[1:0] returns value that was last written. EN2 is reserved and has no influence on the emergency control function of P21.2, should be written with 0.

14.13 Port 22

This section describes the Port 22 functionality.

Port 22 is an 5-bit GPIO port.

14.13.1 Port 22 Registers

The following registers are available on Port 22:

Table 14-17 Port 22 Registers

Register Short Name	Register Long Name	Offset[1] Address	Description see
P22_OUT	Port 22 Output Register	0000_H	**Page 14-101** 2)
P22_OMR	Port 22 Output Modification Register	0004_H	**Page 14-101** 2)
P22_IOCR0	Port 22 Input/Output Control Register 0	0010_H	**Page 14-14**
P22_IOCR4	Port 22 Input/Output Control Register 4	0014_H	**Page 14-102** 2)
P22_IN	Port 22 Input Register	0024_H	**Page 14-101** 2)
P22_PDR0	Port 22 Pad Driver Mode 0 Register	0040_H	**Page 14-103** 2)
P22_ESR	Port 22 Emergency Stop Register	0050_H	**Page 14-103** 2)
P22_OMSR0	Port 22 Output Modification Set Register 0	0070_H	**Page 14-29**
P22_OMSR4	Port 22 Output Modification Set Register 4	0074_H	**Page 14-101** 2)
P22_OMCR0	Port 22 Output Modification Clear Register 0	0080_H	**Page 14-34**
P22_OMCR4	Port 22 Output Modification Clear Register 4	0084_H	**Page 14-101** 2)
P22_OMSR	Port 22 Output Modification Set Register	0090_H	**Page 14-101** 2)
P22_OMCR	Port 22 Output Modification Clear Register	0094_H	**Page 14-101** 2)
P22_ACCEN1	Port 22 Access Enable Register 1	$00F8_H$	**Page 14-43**
P22_ACCEN0	Port 22 Access Enable Register 0	$00FC_H$	**Page 14-41**

1) The absolute addresses are calculated by adding the offset address to the module base address (see **Table 14-3**)

2) These registers are listed and noted here in the Port 22 section because they differ from the general port register description given in **Section 14.3**.

14.13.1.1 Port 22 Output Register

The basic P22_OUT register functionality is described on **Page 14-24**. Port line P22.[15:5] are not connected. Reading P22_OUT bit P[15:8] always return 0, reading P[7:5] returns value that was last written.These connected bits can be also set/reset by the corresponding bits in P22_OMR, P22_OMSR, P22_OMCR, P22_OMSRx, P22_OMCRx (x=0,4).

14.13.1.2 Port 22 Output Modification Register

The basic P22_OMR register functionality is described on **Page 14-26**. However, port lines P22.[15:5] are not connected. The P22_OMR bits PS[7:5] and PCL[7:5], have no direct effect on port lines but only on register bits P22_OUT P[7:5].

14.13.1.3 Port 22 Output Modification Set Register

The basic P22_OMSR register functionality is described on **Page 14-28**. However, port lines P22.[15:5] are not connected.The P22_OMR bits PS[7:5] have no direct effect on port lines but only on register bits P22_OUT P[7:5].

14.13.1.4 Port 22 Output Modification Set Register 4

The basic P22_OMSR4 register functionality is described on **Page 14-30**. However, port lines P22.[7:5] are not connected.The P22_OMR bits PS[7:5] have no direct effect on port lines but only on register bits P22_OUT P[7:5].

14.13.1.5 Port 22 Output Modification Clear Register

The basic P22_OMCR register functionality is described on **Page 14-34**. However, port lines P22.[15:5] are not connected.The P22_OMR bits PCL[7:5], have no direct effect on port lines but only on register bits P22_OUT P[7:5].

14.13.1.6 Port 22 Output Modification Clear Register 4

The basic P22_OMCR4 register functionality is described on **Page 14-35**. However, port lines P22.[7:5] are not connected.The P22_OMR bits PCL[7:5], have no direct effect on port lines but only on register bits P22_OUT P[7:5].

14.13.1.7 Port 22 Input Register

The basic P22_IN register functionality is described on **Page 14-40**. However, port lines P22.[15:5] are not connected. Therefore, bits P[15:5] in register P22_IN are always read as 0.

14.13.1.8 Port 22 Input/Output Control Register 4

The PC[7:5] bit fields in register P22_IOCR4 are not connected.

P22_IOCR4
Port 22 Input/Output Control Register 4

(14_H) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PC7					0			PC6					0		
rw					r			rw					r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC5					0			PC4					0		
rw					r			rw					r		

Field	Bits	Type	Description
PC5, PC6, PC7	[15:11], [23:19], [31:27]	rw	**Reserved** Read as 00000_B after reset; returns value that was written.
PC4	[7:3]	rw	**Port Control for Port 22 Pin 4** (coding see **Table 14-5**)
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

14.13.1.9 Port 22 Pad Driver Mode 0 Register

The basic P22_PDR0 register functionality is described on **Page 14-20**.

P22_PDR0
Port 22 Pad Driver Mode 0 Register (40$_H$) **Reset Value: 3333 3333$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PL7	PD7			PL6	PD6			PL5	PD5			PL4	PD4		
rw	rw			rw	rw			rw	rw			rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PL3	PD3			PL2	PD2			PL1	PD1			PL0	PD0		
rw	rw			rw	rw			rw	rw			rw	rw		

Field	Bits	Type	Description
PD0	[2:0]	rw	**Pad Driver Mode for Port 22 Pin 0**
PD1	[6:4]	rw	**Pad Driver Mode for Port 22 Pin 1**
PD2	[10:8]	rw	**Pad Driver Mode for Port 22 Pin 2**
PD3	[14:12]	rw	**Pad Driver Mode for Port 22 Pin 3**
PD4	[18:16]	rw	**Pad Driver Mode for Port 22 Pin 4**
PD5, PD6, PD7	[22:20], [26:24], [30:28]	rw	**Reserved** Read as 011_B after reset; returns value that was written, must be written with reset value.
PL0, PL1, PL2, PL3, PL4, PL5, PL6, PL7	3, 7, 11, 15, 19, 23, 27, 31	rw	**Reserved** Read as 0, returns values last written, must be written with 0.

14.13.1.10 Port 22 Emergency Stop Register

The basic P22_ESR register functionality is described on **Page 14-38**. Port lines P22.[15:5] are not connected. Reading EN[15:8] always returns 0, reading EN[7:5] returns value that was last written.

14.14 Port 23

This section describes the Port 23 functionality.

14.14.1 Port 23 Configuration

Port 23 is a 1-bit bi-directional general-purpose I/O port.

14.14.2 Port 23 Registers

The following registers are available on Port 23:

Table 14-18 Port 23 Registers

Register Short Name	Register Long Name	Offset[1] Address	Description see
P23_OUT	Port 23 Output Register	0000_H	**Page 14-105** 2)
P23_OMR	Port 23 Output Modification Register	0004_H	**Page 14-108** 2)
P23_IOCR0	Port 23 Input/Output Control Register 0	0010_H	**Page 14-107** 2)
P23_IN	Port 23 Input Register	0024_H	**Page 14-109** 2)
P23_PDR0	Port 23 Pad Driver Mode 0 Register	0040_H	**Page 14-108** 2)
P23_ESR	Port 23 Emergency Stop Register	0050_H	**Page 14-109** 2)
P23_OMSR0	Port 23 Output Modification Set Register 0	0070_H	**Page 14-109** 2)
P23_OMCR0	Port 23 Output Modification Clear Register 0	0080_H	**Page 14-109** 2)
P23_OMSR	Port 23 Output Modification Set Register	0090_H	**Page 14-109** 2)
P23_OMCR	Port 23 Output Modification Clear Register	0094_H	**Page 14-109** 2)
P23_ACCEN1	Port 23 Access Enable Register 1	$00F8_H$	**Page 14-43**
P23_ACCEN0	Port 23 Access Enable Register 0	$00FC_H$	**Page 14-41**

1) The absolute addresses are calculated by adding the offset address to the module base address (see **Table 14-3**)

2) These registers are listed and noted here in the Port 23 section because they differ from the general port register description given in **Section 14.3**.

14.14.2.1 Port 23 Output Register

The basic P23_OUT register functionality is described on **Page 14-24**. Port line P23.0, P23.[15:2] are not connected. Reading the P23_OUT bit P0,P[3:2] returns the value that was last written (0 after reset). Reading the P23_OUT bit P[15:4] always return 0. These

connected bits can be also set/reset by the corresponding bits in P23_OMR, P23_OMSR, P23_OMCR, P23_OMSR0, P23_OMCR0.

14.14.2.2 Port 23 Input/Output Control Register 0

The PC0, PC[3:2] bit fields in register P23_IOCR0 are not connected.

P23_IOCR0
Port 23 Input/Output Control Register 0

(10_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PC3					**0**			**PC2**					**0**		
rw					r			rw					r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC1					**0**			**PC0**					**0**		
rw					r			rw					r		

Field	Bits	Type	Description
PC1	[15:11]	rw	**Port Control for Port 23 Pin 1**
PC0, PC2, PC3	[7:3], [23:19], [31:27]	rw	**Reserved** Read as 00000_B after reset; returns value that was written.
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

14.14.2.3 Port 23 Pad Driver Mode 0 Register

The basic P23_PDR0 register functionality is described on **Page 14-20**.

P23_PDR0
Port 23 Pad Driver Mode 0 Register (40_H) **Reset Value: $0000\ 3333_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PL7	**PD7**			**PL6**	**PD6**			**PL5**	**PD5**			**PL4**	**PD4**		
r	r			r	r			r	r			r	r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PL3	**PD3**			**PL2**	**PD2**			**PL1**	**PD1**			**PL0**	**PD0**		
rw	rw			rw	rw			rw	rw			rw	rw		

Field	Bits	Type	Description
PD1	[6:4]	rw	**Pad Driver Mode for Port 23 Pin 1**
PL1	7	rw	**Pad Level Selection for Port 23 Pin 1**
PD0, PD2, PD3	[2:0], [10:8], [14:12]	rw	**Reserved** Read as 011_B after reset; returns value that was written, must be written with reset value.
PD4, PD5, PD6, PD7	[18:16], [22:20], [26:24], [30:28]	r	**Reserved** Read as 000_B after reset; should be written with 0.
PL0, PL2, PL3	3, 11, 15	rw	**Reserved** Read as 0_B after reset; returns value that was written, must be written with 0.
PL4, PL5, PL6, PL7	19, 23, 27, 31	r	**Reserved** Read as 0_B after reset; should be written with 0.

14.14.2.4 Port 23 Output Modification Register

The basic P23_OMR register functionality is described on **Page 14-26**. However, port lines P23.0, P23.[15:2] are not connected. The P23_OMR bits PS0, PS[3:2] and PCL0, PCL[3:2] have no direct effect on port lines but only on register bits P23_OUT.P0, P[3:2].

14.14.2.5 Port 23 Output Modification Set Register

The basic P23_OMSR register functionality is described on **Page 14-28**. However, port lines P23.0, P23.[15:2] is not connected. The P23_OMSR bits PS0, PS[15:2] have no direct effect on port lines but only on register bits P23_OUT.P0, P[15:2].

14.14.2.6 Port 23 Output Modification Set Register 0

The basic P23_OMSR0 register functionality is described on **Page 14-29**. However, port lines P23.0, P23.[3:2] are not connected. The P23_OMSR0 bits PS0, PS[3:2] have no direct effect on port lines but only on register bits P23_OUT.P0, P[3:2].

14.14.2.7 Port 23 Output Modification Clear Register

The basic P23_OMCR register functionality is described on **Page 14-33**. However, port lines P23.0, P23.[3:2] are not connected. The P23_OMCR bits PCL0, PCL[3:2] have no direct effect on port lines but only on register bits P23_OUT.P0, P[3:2].

14.14.2.8 Port 23 Output Modification Clear Register 0

The basic P23_OMCR0 register functionality is described on **Page 14-34**. However, port lines P23.0, P23.[3:2] are not connected. The P23_OMCR0 bits PCL0, PCL[3:2] have no direct effect on port lines but only on register bits P23_OUT.P0, P[3:2].

14.14.2.9 Port 23 Input Register

The basic P23_IN register functionality is described on **Page 14-40**. However, port lines P23.0, P23.[15:2] are not connected. Therefore, bits P0, P[15:2] in register P23_IN are always read as 0.

14.14.2.10Port 23 Emergency Stop Register

The basic P23_ESR register functionality is described on **Page 14-38**. Port lines P23.0, P23.[15:2] are not connected. Reading the EN0, EN[3:2] returns value that was last written.Reading EN[15:4] always returns 0.

14.15 Port 33

This section describes the Port 33 functionality.

14.15.1 Port 33 Registers

The following registers are available on Port 33:

Table 14-19 Port 00 Registers

Register Short Name	Register Long Name	Offset[1] Address	Description see
P33_OUT	Port 33 Output Register	0000_H	Page 14-46[2]
P33_OMR	Port 33 Output Modification Register	0004_H	Page 14-46[2]
P33_IOCR0	Port 33 Input/Output Control Register 0	0010_H	Page 14-14
P33_IOCR4	Port 33 Input/Output Control Register 4	0014_H	Page 14-15
P33_IOCR8	Port 33 Input/Output Control Register 8	0018_H	Page 14-16
P33_IOCR12	Port 33 Input/Output Control Register 12	$001C_H$	Page 14-47[2]
P33_IN	Port 33 Input Register	0024_H	Page 14-47[2]
P33_PDR0	Port 33 Pad Driver Mode 0 Register	0040_H	Page 14-20
P33_PDR1	Port 33 Pad Driver Mode 1 Register	0044_H	Page 14-48[2]
P33_ESR	Port 33 Emergency Stop Register	0050_H	Page 14-48[2]
P33_OMSR0	Port 33 Output Modification Set Register 0	0070_H	Page 14-29
P33_OMSR4	Port 33 Output Modification Set Register 4	0074_H	Page 14-30
P33_OMSR8	Port 33 Output Modification Set Register 8	0078_H	Page 14-31
P33_OMSR12	Port 33 Output Modification Set Register 12	$007C_H$	Page 14-46[2]
P33_OMCR0	Port 33 Output Modification Clear Register 0	0080_H	Page 14-34
P33_OMCR4	Port 33 Output Modification Clear Register 4	0084_H	Page 14-35
P33_OMCR8	Port 33 Output Modification Clear Register 8	0088_H	Page 14-36
P33_OMCR12	Port 33 Output Modification Clear Register 12	$008C_H$	Page 14-46[2]
P33_OMSR	Port 33 Output Modification Set Register	0090_H	Page 14-46[2]
P33_OMCR	Port 33 Output Modification Clear Register	0094_H	Page 14-46[2]
P33_ACCEN1	Port 33 Access Enable Register 1	$00F8_H$	Page 14-43
P33_ACCEN0	Port 33 Access Enable Register 0	$00FC_H$	Page 14-41

1) The absolute addresses are calculated by adding the offset address to the module base address (see Table 14-3)

2) These registers are listed and noted here in the Port 33 section because they differ from the general port register description given in Section 14.3.

14.15.1.1 Port 33 Output Register

The basic P33_OUT register functionality is described on **Page 14-24**. Port lines P33.[15:13] are not connected. Reading the P33_OUT bits P[15:13] returns the value that was last written. These connected bits can be also set/reset by the corresponding bits in P33_OMR, P33_OMSR, P33_OMCR, P33_OMSRx (x=0,4,8,12), P33_OMCRx (x=0,4,8,12).

14.15.1.2 Port 33 Output Modification Register

The basic P33_OMR register functionality is described on **Page 14-26**. However, port lines P33.[15:13] are not connected. The P33_OMR bits PS[15:13] and PCL[15:13] have no direct effect on port lines but only on register bits P33_OUT.P[15:13].

14.15.1.3 Port 33 Output Modification Set Register

The basic P33_OMSR register functionality is described on **Page 14-28**. However, port lines P33.[15:13] are not connected. The P33_OMSR bits PS[15:13] have no direct effect on port lines but only on register bits P33_OUT.P[15:13].

14.15.1.4 Port 33 Output Modification Set Register 12

The basic P33_OMSR12 register functionality is described on **Page 14-32**. However, port lines P33.[15:13] are not connected. The P33_OMSR12 bits PS[15:13] have no direct effect on port lines but only on register bits P33_OUT.P[15:13].

14.15.1.5 Port 33 Output Modification Clear Register

The basic P33_OMCR register functionality is described on **Page 14-33**. However, port lines P33.[15:13] are not connected. The P33_OMCR bits PCL[15:13] have no direct effect on port lines but only on register bits P33_OUT.P[15:13].

14.15.1.6 Port 33 Output Modification Clear Register 12

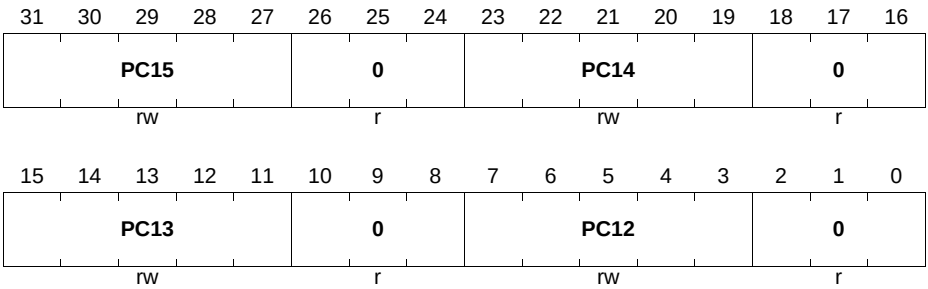
The basic P33_OMCR12 register functionality is described on **Page 14-37**. However, port lines P33.[15:13] are not connected. The P33_OMCR12 bits PCL[15:13] have no direct effect on port lines but only on register bits P33_OUT.P[15:13].

14.15.1.7 Port 33 Input/Output Control Register 12

The PC13, PC14 and PC15 bit fields in register P33_IOCR12 are not connected.

P33_IOCR12
Port 33 Input/Output Control Register 12

($1C_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
PC13	[15:11]	rw	**Reserved** Read as 00000_B after reset; returns value that was written.
PC14	[23:19]	rw	**Reserved** Read as 00000_B after reset; returns value that was written.
PC15	[31:27]	rw	**Reserved** Read as 00000_B after reset; returns value that was written.
PC12	[7:3]	rw	**Port Control for Port 33.12** (coding see **Table 14-5**)
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

14.15.1.8 Port 33 Input Register

The basic P33_IN register functionality is described on **Page 14-40**. However, port lines P33.[15:13] are not connected. Therefore, bits P[15:13] in register P33_IN are always read as 0.

14.15.1.9 Port 33 Pad Driver Mode 1 Register

P33_PDR1
Port 33 Pad Driver Mode 1 Register (44$_H$) **Reset Value: 3333 3333$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PL15		PD15		PL14		PD14		PL13		PD13		PL12		PD12	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PL11		PD11		PL10		PD10		PL9		PD9		PL8		PD8	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
PD8	[2:0]	rw	**Pad Driver Mode for Port 33 Pin 8**
PD9	[6:4]	rw	**Pad Driver Mode for Port 33 Pin 9**
PD10	[10:8]	rw	**Pad Driver Mode for Port 33 Pin 10**
PD11	[14:12]	rw	**Pad Driver Mode for Port 33 Pin 11**
PD12	[18:16]	rw	**Pad Driver Mode for Port 33 Pin 12**
PD13, PD14, PD15	[22:20], [26:24], [30:28]	rw	**Reserved** Read as 011$_B$ after reset; returns value that was written, must be written with reset value.
PL8, PL9, PL10, PL11, PL12, PL13, PL14, PL15	3, 7, 11, 15, 19, 23, 27, 31	rw	**Reserved** Read as 0, returns values last written, must be written with 0.

14.15.1.10Port 33 Emergency Stop Register

The basic P33_ESR register functionality is described on **Page 14-38**. Port lines P33.[15:13] are not connected. Reading the P33_ESR bits EN[15:13] returns the value that was last written. EN8 is reserved and has no influence on the emergency control function of P33.8, should be written with 0.

14.16 Port 34

This section describes the Port 34 functionality.

Port 34 is an 4-bit GPIO port.

14.16.1 Port 34 Registers

The following registers are available on Port 34:

Table 14-20 Port 13 Registers

Register Short Name	Register Long Name	Offset[1] Address	Description see
P34_OUT	Port 34 Output Register	0000_H	Page 14-70[2]
P34_OMR	Port 34 Output Modification Register	0004_H	Page 14-70[2]
P34_IOCR0	Port 34 Input/Output Control Register 0	0010_H	Page 14-14
P34_IN	Port 34 Input Register	0024_H	Page 14-71[2]
P34_PDR0	Port 34 Pad Driver Mode 0 Register	0040_H	Page 14-72[2]
P34_ESR	Port 34 Emergency Stop Register	0050_H	Page 14-72[2]
P34_OMSR0	Port 34 Output Modification Set Register 0	0070_H	Page 14-29
P34_OMCR0	Port 34 Output Modification Clear Register 0	0080_H	Page 14-34
P34_OMSR	Port 34 Output Modification Set Register	0090_H	Page 14-71[2]
P34_OMCR	Port 34 Output Modification Clear Register	0094_H	Page 14-71[2]
P34_ACCEN1	Port 34 Access Enable Register 1	$00F8_H$	Page 14-43
P34_ACCEN0	Port 34 Access Enable Register 0	$00FC_H$	Page 14-41

1) The absolute addresses are calculated by adding the offset address to the module base address (see Table 14-3)

2) These registers are listed and noted here in the Port 34 section because they differ from the general port register description given in Section 14.3.

14.16.1.1 Port 34 Output Register

The basic P34_OUT register functionality is described on Page 14-24. Port line P34.[15:4] are not connected. Reading the P34_OUT bit P[15:4] always return 0. These connected bits can be also set/reset by the corresponding bits in P34_OMR, P34_OMSR, P34_OMCR, P34_OMSR0, P34_OMCR0.

14.16.1.2 Port 34 Output Modification Register

The basic P34_OMR register functionality is described on Page 14-26. However, port lines P34.[15:4] are not connected.

14.16.1.3 Port 34 Output Modification Set Register

The basic P34_OMSR register functionality is described on **Page 14-28**. However, port lines P34.[15:4] are not connected.

14.16.1.4 Port 34 Output Modification Clear Register

The basic P34_OMCR register functionality is described on **Page 14-33**. However, port lines P34.[15:4] are not connected.

14.16.1.5 Port 34 Input Register

The basic P34_IN register functionality is described on **Page 14-40**. However, port lines P34.[15:4] are not connected. Therefore, bits P[15:4] in register P34_IN are always read as 0.

14.16.1.6 Port 34 Pad Driver Mode 0 Register

The basic P34_PDR0 register functionality is described on **Page 14-20**.

P34_PDR0
Port 34 Pad Driver Mode 0 Register (40_H) **Reset Value: $0000\ 3333_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PL7	**PD7**			**PL6**	**PD6**			**PL5**	**PD5**			**PL4**	**PD4**		
r	r			r	r			r	r			r	r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PL3	**PD3**			**PL2**	**PD2**			**PL1**	**PD1**			**PL0**	**PD0**		
rw	rw			rw	rw			rw	rw			rw	rw		

Field	Bits	Type	Description
PD0, PD1, PD2, PD3	[2:0], [6:4], [10:8], [14:12]	rw	**Pad Driver Mode for Port 34 Pin 0 to 3**
PL0, PL1, PL2, PL3	3, 7, 11, 15	rw	**Reserved** Read as 0, returns values last written, must be written with 0.
PD4, PD5, PD6, PD7	[18:16], [22:20], [26:24], [30:28]	r	**Reserved** Read as 000_B after reset; should be written with 0.
PL4, PL5, PL6, PL7	19, 23, 27, 31	r	**Reserved** Read as 0_B after reset; should be written with 0.

14.16.1.7 Port 34 Emergency Stop Register

The basic P34_ESR register functionality is described on **Page 14-38**. Port lines P34.[15:4] are not connected. Reading the P34_ESR bits EN[15:4] always returns 0.

14.17 Port 40

This section describes the Port 40 functionality in detail.

14.17.1 Port 40 Configuration

Port 40 is a 12-bit input port.

14.17.2 Port 40 Registers

The following registers are available on Port 40:

Table 14-21 Port 40 Registers

Register Short Name	Register Long Name	Offset [1] Address	Description see
P40_IOCR0	Port 40 Input/Output Control Register 0	0010_H	Page 14-121 [2]
P40_IOCR4	Port 40 Input/Output Control Register 4	0014_H	Page 14-121 [2]
P40_IOCR8	Port 40 Input/Output Control Register 8	0018_H	Page 14-121 [2]
P40_IN	Port 40 Input Register	0024_H	Page 14-40
P40_PDISC	Port 40 Pin Function Decision Control Register	0060_H	Page 14-123 [2]
P40_PCSR	Port 40 Pin Controller Select Register	0064_H	Page 14-126 [2]
P40_ACCEN1	Port 40 Access Enable Register 1	$00F8_H$	Page 14-43 [2]
P40_ACCEN0	Port 40 Access Enable Register 0	$00FC_H$	Page 14-41 [2]

1) The absolute addresses are calculated by adding the offset address to the module base address (see **Table 14-3**)

2) These registers are listed here in the Port 40 section because they differ from the general port register description given in **Section 14.3**.

14.17.3 Port 40 Input/Output Control Registers

P40_IOCR0
Port 40 Input/Output Control Register 0

(10_H) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PC3					0			PC2					0		
rw					r			rw					r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC1					0			PC0					0		
rw					r			rw					r		

Field	Bits	Type	Description
PC0, PC1, PC2, PC3	[7:3], [15:11], [23:19], [31:27]	rw	**Port Control for Port n Pin 0 to 3** This bit field defines the Port n line x functionality according to **Table 14-5**, only input selection apply.
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

P40_IOCR4
Port 40 Input/Output Control Register 4

(14_H) **Reset Value: 0000 0000$_H$**

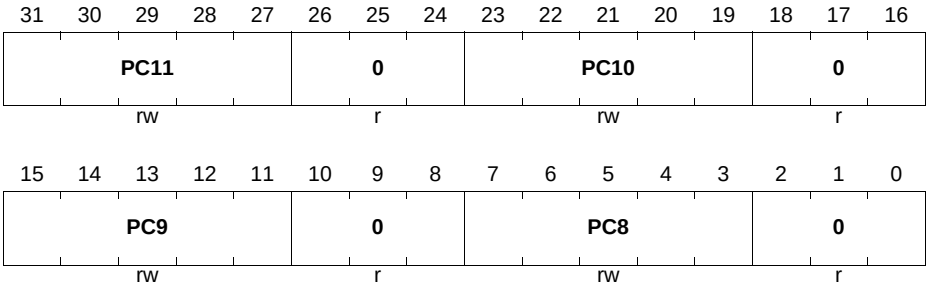
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PC7					0			PC6					0		
rw					r			rw					r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC5					0			PC4					0		
rw					r			rw					r		

Field	Bits	Type	Description
PC4, PC5, PC6, PC7	[7:3], [15:11], [23:19], [31:27]	rw	**Port Control for Port n Pin 4 to 7** This bit field defines the Port n line x functionality according to **Table 14-5**, only input selection apply.
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

P40_IOCR8
Port 40 Input/Output Control Register 8

(18_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
PC8, PC9, PC10, PC11	[7:3], [15:11], [23:19], [31:27]	rw	**Port Control for Port 40 Pin 8 to 11** (coding see **Table 14-5**, only input selection apply.)
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

14.17.4 Port 40 Pin Function Decision Control Register

P40_PDISC

Port 40 Pin Function Decision Control Register(60_H) **Reset Value: 0000 0FFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	PDIS 11	PDIS 10	PDIS 9	PDIS 8	PDIS 7	PDIS 6	PDIS 5	PDIS 4	PDIS 3	PDIS 2	PDIS 1	PDIS 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PDIS0	0	rw	**Pin Function Decision Control for Pin 0** The bit selects the function of P40.0 The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 0. 1_B Pad is used for ADC analog input x.
PDIS1	1	rw	**Pin Function Decision Control for Pin 1** The bit selects the function of P40.1 The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 1. 1_B Pad is used for ADC analog input x.
PDIS2	2	rw	**Pin Function Decision Control for Pin 2** The bit selects the function of P40.2. The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 2 1_B Pad is used for ADC analog input x.
PDIS3	3	rw	**Pin Function Decision Control for Pin 3** The bit selects the function of P40.3. The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 3. 1_B Pad is used for ADC analog input x.

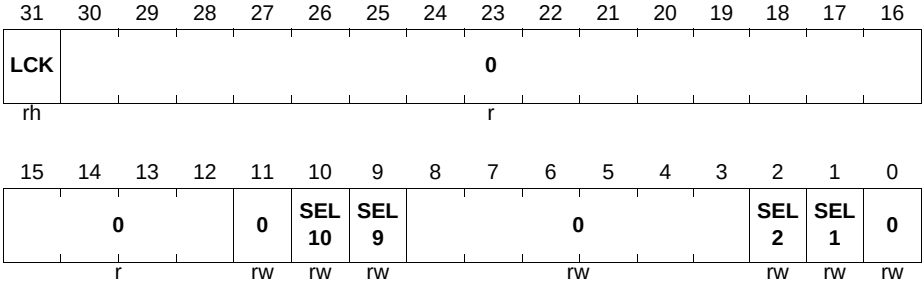
Field	Bits	Type	Description
PDIS4	4	rw	**Pin Function Decision Control for Pin 4** The bit selects the function of P40.4. The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 4. 1_B Pad is used for ADC analog input x.
PDIS5	5	rw	**Pin Function Decision Control for Pin 5** The bit selects the function of P40.5. The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 5. 1_B Pad is used for ADC analog input x.
PDIS6	6	rw	**Pin Function Decision Control for Pin 6** The bit selects the function of P40.6. The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 6. 1_B Pad is used for ADC analog input x.
PDIS7	7	rw	**Pin Function Decision Control for Pin 7** The bit selects the function of P40.7. The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 7. 1_B Pad is used for ADC analog input x.
PDIS8	8	rw	**Pin Function Decision Control for Pin 8** The bit selects the function of P40.8. The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 8. 1_B Pad is used for ADC analog input x.
PDIS9	9	rw	**Pin Function Decision Control for Pin 9** The bit selects the function of P40.9. The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 9. 1_B Pad is used for ADC analog input x.
PDIS10	10	rw	**Pin Function Decision Control for Pin 10** The bit selects the function of P40.10. The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 10. 1_B Pad is used for ADC analog input x.

Field	Bits	Type	Description
PDIS11	11	rw	**Pin Function Decision Control for Pin 11** The bit selects the function of P40.11. The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 11. 1_B Pad is used for ADC analog input x.
0	[15:12]	rw	**Reserved** Read as 0; should be written with 0.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

14.17.5 Port 40 Pin Controller Select Register

P40_PCSR enables or disables the VADC Multiplexer Diagnostics (MD) feature.

P40_PCSR
Port 40 Pin Controller Select Register (64_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
SELx (x = 2-1)	0 + x	rw	**Pin Controller Select for Pin x** This bit enables or disables the VADC Multiplexer Diagnostics (MD) feature. For the respective analog signal connection, see analog connections table in VADC chapter. 0_B Disables VADC MD feature. 1_B Enables VADC MD feature.
SELx (x = 10-9)	0 + x	rw	**Pin Controller Select for Pin x** This bit enables or disables the VADC Multiplexer Diagnostics (MD) feature. For the respective analog signal connection, see analog connections table in VADC chapter. 0_B Disables VADC MD feature. 1_B Enables VADC MD feature.
LCK	31	rh	**Lock Status** This bit indicates if the register can be updated with a new value or if the register is locked and a write action from the bus has no effect. 0_B The register is unlocked and can be updated. 1_B The register is locked and can not be updated.

Field	Bits	Type	Description
0	0, [8:3], 11	rw	**Reserved** Read as 0; returns value last written, must be written with 0.
0	[30:16], [15:12]	r	**Reserved** Read as 0; should be written with 0.

14.18 Port 41

This section describes the Port 41 functionality in detail.

14.18.1 Port 41 Configuration

Port 41 is a 12-bit input port.

14.18.2 Port 41 Registers

The following registers are available on Port 41:

Table 14-22 Port 40 Registers

Register Short Name	Register Long Name	Offset [1] Address	Description see
P41_IOCR0	Port 41 Input/Output Control Register 0	0010_H	Page 14-130 [2]
P41_IOCR4	Port 41 Input/Output Control Register 4	0014_H	Page 14-130 [2]
P41_IOCR8	Port 41 Input/Output Control Register 8	0018_H	Page 14-130 [2]
P41_IN	Port 41 Input Register	0024_H	Page 14-40
P41_PDISC	Port 41 Pin Function Decision Control Register	0060_H	Page 14-132 [2]
P41_PCSR	Port 41 Pin Controller Select Register	0064_H	Page 14-132 [2]
P41_ACCEN1	Port 41 Access Enable Register 1	$00F8_H$	Page 14-43 [2]
P41_ACCEN0	Port 41 Access Enable Register 0	$00FC_H$	Page 14-41 [2]

1) The absolute addresses are calculated by adding the offset address to the module base address (see **Table 14-3**)

2) These registers are listed here in the Port 41 section because they differ from the general port register description given in **Section 14.3**.

14.18.3 Port 41 Input/Output Control Registers

P41_IOCR0
Port 41 Input/Output Control Register 0

(10_H) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PC3					0			PC2					0		
rw					r			rw					r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC1					0			PC0					0		
rw					r			rw					r		

Field	Bits	Type	Description
PC0, PC1, PC2, PC3	[7:3], [15:11], [23:19], [31:27]	rw	**Port Control for Port n Pin 0 to 3** This bit field defines the Port n line x functionality according to **Table 14-5**, only input selection apply.
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

P41_IOCR4
Port 41 Input/Output Control Register 4

(14_H) **Reset Value: 0000 0000$_H$**

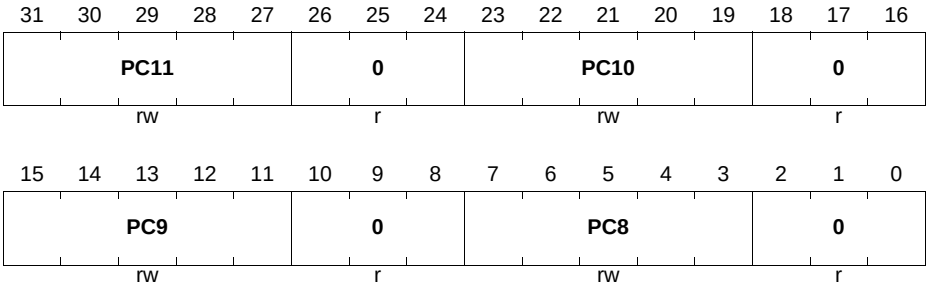
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PC7					0			PC6					0		
rw					r			rw					r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC5					0			PC4					0		
rw					r			rw					r		

Field	Bits	Type	Description
PC4, PC5, PC6, PC7	[7:3], [15:11], [23:19], [31:27]	rw	**Port Control for Port n Pin 4 to 7** This bit field defines the Port n line x functionality according to **Table 14-5**, only input selection apply.
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

P41_IOCR8
Port 41 Input/Output Control Register 8

(18_H) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
PC8, PC9, PC10, PC11	[7:3], [15:11], [23:19], [31:27]	rw	**Port Control for Port 41 Pin 8 to 11** (coding see **Table 14-5**, only input selection apply.)
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

14.18.4 Port 41 Pin Function Decision Control Register

P41_PDISC

Port 41 Pin Function Decision Control Register(60_H) **Reset Value: 0000 0FFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	PDIS 11	PDIS 10	PDIS 9	PDIS 8	PDIS 7	PDIS 6	PDIS 5	PDIS 4	PDIS 3	PDIS 2	PDIS 1	PDIS 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PDIS0	0	rw	**Pin Function Decision Control for Pin 0** The bit selects the function of P41.0 The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 0. 1_B Pad is used for ADC analog input x.
PDIS1	1	rw	**Pin Function Decision Control for Pin 1** The bit selects the function of P41.1 The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 1. 1_B Pad is used for ADC analog input x.
PDIS2	2	rw	**Pin Function Decision Control for Pin 2** The bit selects the function of P41.2. The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 2 1_B Pad is used for ADC analog input x.
PDIS3	3	rw	**Pin Function Decision Control for Pin 3** The bit selects the function of P41.3. The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 3. 1_B Pad is used for ADC analog input x.

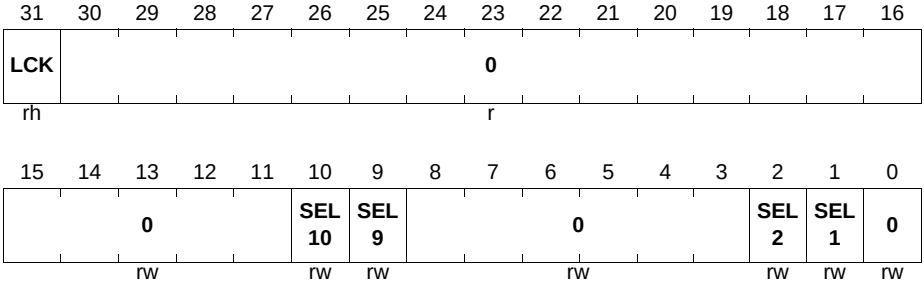
Field	Bits	Type	Description
PDIS4	4	rw	**Pin Function Decision Control for Pin 4** The bit selects the function of P41.4. The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 4. 1_B Pad is used for ADC analog input x.
PDIS5	5	rw	**Pin Function Decision Control for Pin 5** The bit selects the function of P41.5. The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 5. 1_B Pad is used for ADC analog input x.
PDIS6	6	rw	**Pin Function Decision Control for Pin 6** The bit selects the function of P41.6. The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 6. 1_B Pad is used for ADC analog input x.
PDIS7	7	rw	**Pin Function Decision Control for Pin 7** The bit selects the function of P41.7. The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 7. 1_B Pad is used for ADC analog input x.
PDIS8	8	rw	**Pin Function Decision Control for Pin 8** The bit selects the function of P41.8. The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 8. 1_B Pad is used for ADC analog input x.
PDIS9	9	rw	**Pin Function Decision Control for Pin 9** The bit selects the function of P41.9. The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 9. 1_B Pad is used for ADC analog input x.
PDIS10	10	rw	**Pin Function Decision Control for Pin 10** The bit selects the function of P41.10. The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 10. 1_B Pad is used for ADC analog input x.

Field	Bits	Type	Description
PDIS11	11	rw	**Pin Function Decision Control for Pin 11** The bit selects the function of P41.11. The default state of the pad selects the ADC analog input. 0_B Pad is used for digital input 11. 1_B Pad is used for ADC analog input x.
0	[15:12]	rw	**Reserved** Read as 0; should be written with 0.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

14.18.5 Port 41 Pin Controller Select Register

P41_PCSR enables or disables the VADC Multiplexer Diagnostics (MD) feature.

P41_PCSR
Port 41 Pin Controller Select Register (64_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
SELx (x = 2-1)	0 + x	rw	**Pin Controller Select for Pin x** This bit enables or disables the VADC Multiplexer Diagnostics (MD) feature. For the respective analog signal connection, see analog connections table in VADC chapter. 0_B Disables VADC MD feature. 1_B Enables VADC MD feature.
SELx (x = 10-9)	0 + x	rw	**Pin Controller Select for Pin x** This bit enables or disables the VADC Multiplexer Diagnostics (MD) feature. For the respective analog signal connection, see analog connections table in VADC chapter. 0_B Disables VADC MD feature. 1_B Enables VADC MD feature.
LCK	31	rh	**Lock Status** This bit indicates if the register can be updated with a new value or if the register is locked and a write action from the bus has no effect. 0_B The register is unlocked and can be updated. 1_B The register is locked and can not be updated.

Field	Bits	Type	Description
0	0, [8:3], [15:11]	rw	**Reserved** Read as 0; returns value last written, must be written with 0.
0	[30:16]	r	**Reserved** Read as 0; should be written with 0.

14.19 Register Behavior for Unavailable Port Pins

Table 14-23 Register Behavior of unavailable pins

Register	Behavior
Pn_OUT, Pn_ESR, Pn_PDR0, Pn_PDR1, Pn_IOCRx (x=0-3), Pn_PDISC	The unused control fields within the same nibble pin group as available pins are "rw", returns value that was last written and must be written with reset value. Otherwise, these unused control fields are 'r', read as 0 and should be written with reset value.
Pn_OMR, Pn_OMSR, Pn_OMSRx, Pn_OMCR, Pn_OMCRx, (x=0-3)	The unused control fields within the same nibble pin group as available pins are able to control Pn_OUT. Otherwise, these unused control fields do not have an influence on Pn_OUT.

15 Direct Memory Access (DMA)

This chapter describes the DMA Controller. It contains the following sections:

- Features (see **Section 15.2**)
- Block Diagram (see **Section 15.3**)
- Functional Description (see **Section 15.4**)
- Power Modes (see **Section 15.5**)
- Functional Safety Features (see **Section 15.6**)
- Debug Features (see **Section 15.7**)
- Register Description (see **Section 15.8**)
- Use Cases (see **Section 15.9**)

*Note: The DMA kernel register names described in **Section 15.8** are referenced in the TC21x/TC22x/TC23x User's Manual by the module name prefix "DMA_".*

15.1 What is new

Major differences of the RAM based AURIX DMA compared to the register based AURIX DMA are

- Any DMA move engine can service a DMA request from any DMA channel. There is no fixed mapping of DMA channels to move engines. DMA requests from the highest number DMA channel are serviced first by the DMA controller. The DMA move engine is the module that executes DMA channel requests.
- The DMA channel transaction control set is stored in the DMARAM.The move engines uses a copy of the DMARAM channel transaction control set to service DMA requests.
- DMA channels can be assigned to one of four hardware resource partitions that determine access protection of the DMARAM channel transaction control set.
- DMA channels can be daisy chained. On completing a DMA transaction a DMA channel can initiate a DMA transaction in the immediate next lower priority channel.
- DMA channel source and address pointers are 32-bit wide address counters. The source and destination wrap buffers are optionally selectable.
- Support for DMA Linked Lists. The current DMA transaction can load the next DMA transaction control set into the DMARAM by overwriting the existing channel transaction control set. It can optionally auto start the next DMA transaction.
- Support for DMA Double Buffering. The DMA transaction can execute read or fill DMA transfer from one of two source or destination buffers. A control bit allows the re-direction of DMA transfers from the one buffer to the other buffer.
- All DMA channels support safe DMA operation by the addition of logic to calculate checksums during DMA Moves.
- The move engine supports 64-bit data transfers to SRI addresses. Each move engine has its own 256-bit read buffer to access resources in the cached address ranges.
- The DMA kernel is clocked at the SRI clock frequency.

15.2 Features

The DMA controller is a fast and flexible DMA controller that has the following features:

- The DMA controller supports 16 DMA channels:
  - DMA channel 015 has the highest priority.
  - DMA channel 000 has the lowest priority.
- DMA channel hardware requests are sourced from the Interrupt Router (IR) Interrupt Control Unit (ICU). The system architecture defines a dedicated ICU to arbitrate between peripheral interrupt triggers and generate a DMA hardware request.
  - Any peripheral that can trigger an interrupt can initiate a DMA transfer.
- DMA channel software requests.
- The DMA controller supports 2 move engines for the parallel execution of DMA requests:
  - 1 X DMA sub-block (move engine) services 1 X active DMA channel.
  - DMA sub-blocks (move engines) work in parallel.
- The DMA can be hardware configured to support three programmable levels of System Peripheral Bus (SPB) bus priority through the single on chip bus data path to support the optional addition of a Debug Interface.
  - Setting high priority supports enhanced debug access. The debugger wins arbitration at the internal DMA bus switch and over all other bus master requests in order to gain immediate access to the SPB. There is a potential impact on performance.
  - Setting low priority supports non intrusive debugging. The debugger only wins arbitration when no other DMA access requesters or bus masters are requesting access to the bus. There is minimal performance impact.
- Individually programmable operation modes for each DMA channel
- Full 32-bit addressing capability of each DMA channel
  - 4 Gbyte address range.
  - Circular buffer addressing mode with flexible circular buffer sizes
- Data block move throughput
  - DMA transaction moving data from an SRI-source to an SRI-destination supports >8 Mbyte moves per DMA transaction.
  - DMA transaction moving data either from an FPI-source or to an FPI-destination supports >1 Mbyte moves per DMA transaction.
- The DMA transaction control set is stored in the DMARAM.
- DMA read move and DMA write moves are directed by the DMA switch to different sources and destinations depending on the source or destination address.
- Buffer capability for move actions on the buses (at least 1 move per bus is buffered)
- Programmable data width of DMA moves:
  - SPB master interface: 8-bit, 16-bit, or 32-bit.
  - SRI master interface: 8-bit, 16-bit, 32-bit, 64-bit, 128-bit or 256-bit.
- Interrupt Triggers:

  - Each DMA channel generates one traffic management interrupt trigger with an unique interrupt vector and priority level.
  - The DMA generates one error interrupt trigger.
- Operating frequencies:
  - The DMA controller kernel (active channel, move engine and SRI master interface) work at the SRI clock frequency in order to maximise the data throughput for DMA moves from SRI source addresses to SRI destination addresses.
  - The DMA controller configuration sector (slave interface, DMARAM, ICU interface and SPB Master interface) work at the SPB clock frequency.
- Slave based access protection:
  - DMA channel transaction control sets are access protected.
  - Each DMA channel is assigned to a hardware resource partition.
  - Access control to a hardware partition is via enabling of individual Master TAG IDs to have write access to the hardware partition.
  - Each DMA hardware resource partition has an unique Master TAG IDs and is configured to make accesses in supervisor or user mode.

15.3 Block Diagram

The DMA Controller transfers data from data source locations to data destination locations without intervention of the CPU or other on chip devices. One data move operation is controlled by an active DMA channel. A DMA sub-block can service DMA requests from any of the DMA channels. The Bus Switch provides the connection between a DMA sub-block and on chip bus interfaces.

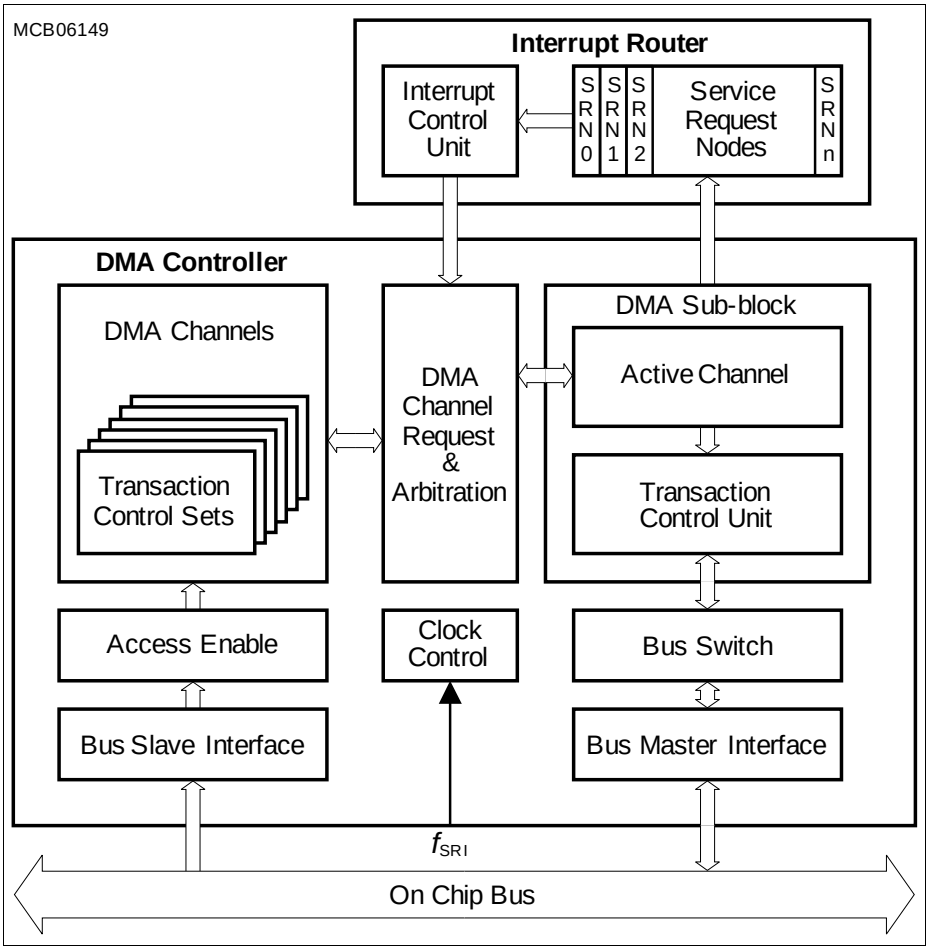


Figure 15-1 DMA Block Diagram

Programming of the DMA transaction control sets is via a bus slave interface. The slave interface supports clock control and access protection to the DMA controller. The Access

Enable registers are safe endinit protected and support register monitoring safety measures.

Address decoding and DMA request wiring are managed outside the DMA controller.

15.4 Functional Description

15.4.1 Definition of Terms

Some basic terms must be defined for the functional description of the DMA controller.

DMA Move

A DMA move is an operation that always consists of two parts:

1. A read move that loads data from a data source into the DMA controller
2. A write move that puts data from the DMA controller to a data destination

Within a DMA move, data is always moved from the data source via the DMA controller to the data destination. Data is temporarily stored in the DMA controller. The data widths of read move and write move are typically identical (8-bit, 16-bit, 32-bit, 64-bit, 128-bit or 256-bit).

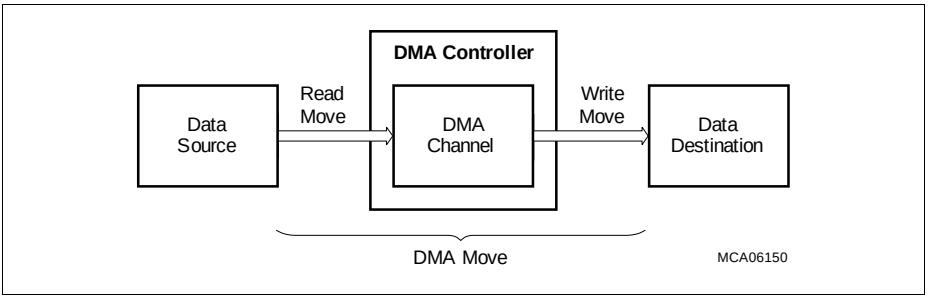


Figure 15-2 DMA Definition of Terms

DMA Transfer

A DMA transfer can be composed of 1, 2, 3, 4, 5, 8, 9 or 16 DMA moves.

DMA Transaction

A DMA transaction is composed of several (at least one) DMA transfers. The Transfer Count determines the number of DMA transfers within one DMA transaction.

Example:
1024 word (32-bit wide) transactions can be composed of 256 transfers of four DMA word moves, or 128 transfers of eight DMA word moves.

Linked List

A linked list is a series of DMA transactions executed in the same DMA channel.

15.4.2 DMA Principles

The DMA controller supports DMA moves from one address location to another location.

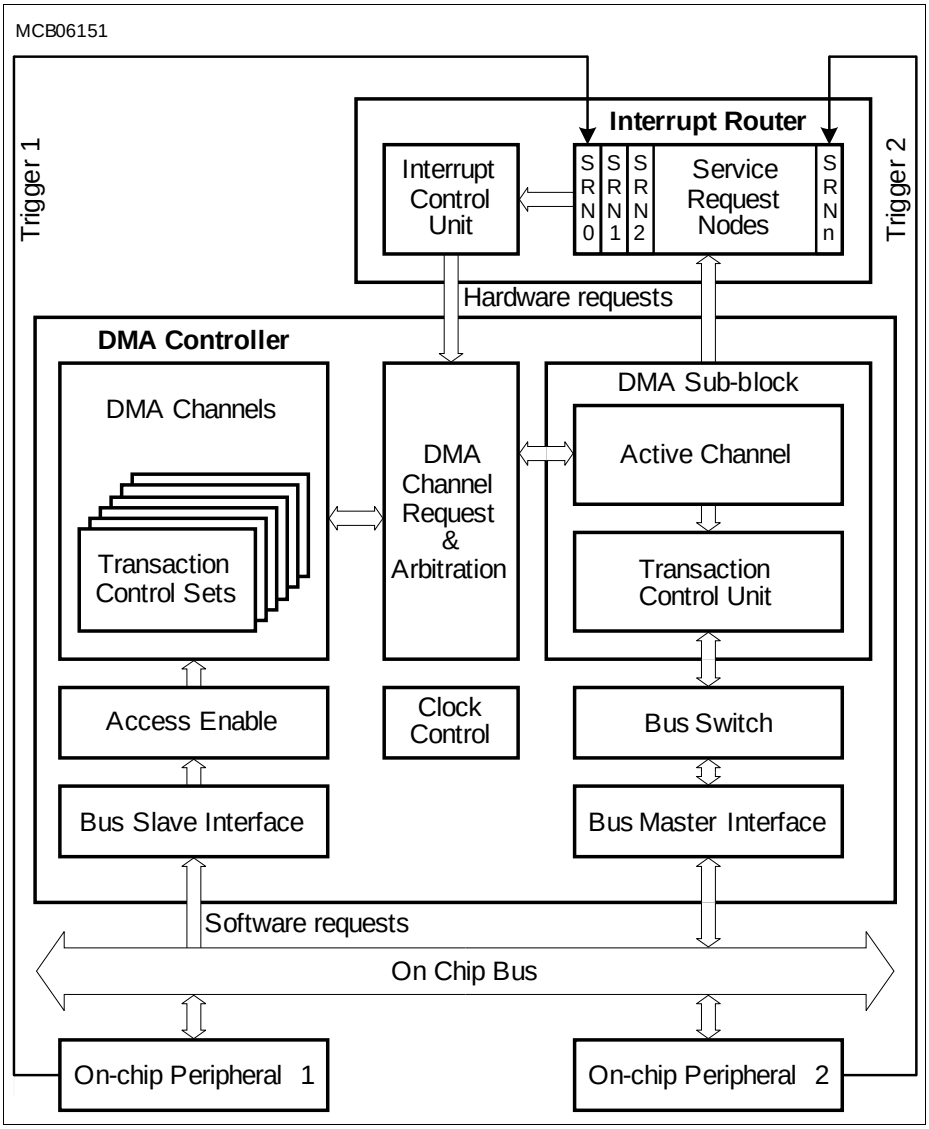


Figure 15-3 DMA System Overview

DMA moves can be requested either by hardware or by software. DMA hardware requests are triggered by specific request lines from the Interrupt Control Unit (ICU) or from other DMA channels (see **Figure 15-3**). Typically, the parallel occurrence of DMA requests and interrupts requests for DMA channels is possible. Therefore, the ICU and the DMA controller can react independently to interrupt DMA requests that have been generated by one source.

The DMA controller primarily consists of DMA channels, sub-blocks (move engines) and a bus switch. Once configured, the DMA sub-blocks are able to act as a master on the SPB Bus and on the SRI Bus.

15.4.3 DMA Channel Functionality

The transaction control set of each DMA channel is stored in the DMARAM in an array of 8 X 32-bit words. When a DMA request is serviced the associated transaction control set is copied from the DMARAM to the active DMA channel register set within a DMA sub-block. The transaction control set of a DMA channel includes functionality to support the calculation of Cyclic Redundancy Checker (CRC) checksums for source and destination addresses and read data to support enhanced data integrity checking. The index “z” refers to the channel number (z = 000 to a maximum of z = 015) within the DMA controller.

DMA channels are associated with a hardware resource partition.The index “y” refers to the hardware resource number (y = 0-3). The move engine and error registers located in the DMA sub-block use the index “x” to refer to the DMA sub-block number (x = 0-1).

When a DMA sub-block x services a DMA request from DMA channel z then the transaction control set (8 X 32-bit words) is copied from the DMARAM to the following active channel register set in DMA sub-block x:

- Channel x Status Register MExCHSR (for details, see **Page 15-108**)
- Channel x Control Register MExCHCR (for details, see **Page 15-112**)
- Channel x Address and Interrupt Control Register MExADICR (for details, see **Page 15-116**)
- Channel x Shadow Address Register MExSHADR (for details, see **Page 15-111**)
- Channel x Destination Address Register MExDADR (for details, see **Page 15-126**)
- Channel x Source Address Register MExSADR (for details, see **Page 15-127**)
- Channel x Source and Destination Address CRC Register MExSDCRC (for details, see **Page 15-128**)
- Channel x Read Data CRC Register MExRDCRC (for details, see **Page 15-129**)

15.4.3.1 Shadowed Source or Destination Address

As a typical application, an ASC module that receives data (fixed source address) has to deliver it to a memory buffer using a DMA transaction (variable destination address). After a certain amount of data has been transferred, a new DMA transaction should be

initiated to deliver further ASC data into another memory buffer. While the destination address register is updated during a running DMA transaction with the actual destination address, a shadow mechanism allows programming of a new destination address without disturbing the content of the destination address register. In this case, the new destination address is written into a buffer register, i.e. the shadow address register. At the start of the next DMA transaction, the new address is transferred from this shadow address register to the destination address register without CPU intervention. This shadow mechanism avoids the CPU having to check for the end of a DMA transaction before reprogramming address registers.

The shadow address register can be used also to store a source address. However, it cannot store source and destination address at the same time. This means that the shadow mechanism makes it possible to automatically update either a new source address, or a new destination address at the start of a DMA transaction. If both address registers (for source and destination address) have to be updated for the next DMA transaction, a running DMA transaction for this channel must be finished. After that, source and destination address registers can be written before the next DMA transaction is started.

Only one address register can be shadowed while a transaction is running, because the shadow register can only be assigned either to the source or to the destination address. Note that the shadow address transfer mechanism has the same behavior in Single and Continuous Mode.

When the shadow mechanism is disabled the DMA sub-block active channel shadow address register MExSHADR is always read as 0000 0000_H.

DMA Channel Idle

If no DMA transaction is running and a new source or destination address is written to DMA channel z then the new address value is directly written into the DMARAM source address word (SADRz) or destination address word (DADRz). Writes to the shadow address word (SHADRz) are as follows:

- Read Only Mode: the shadow address is not writable (ADICRz.SHCT = 0001_B or 0010_B) and a write to the shadow address word SHADRz will result in a bus error.
- Direct Write Mode: the shadow address is directly writable (ADICRz.SHCT = 0101_B or 0110_B) and the shadow address word SHADRz is updated.

DMA Channel Active

In the case when move engine x is servicing a DMA request from DMA channel z and a DMA transfer is in progress then address updates are dependent on the shadow control settings:

- Read Only Mode: **Figure 15-4** shows the actions that take place when a source address register is updated (destination register update happens in an equivalent

manner). The shadow address is not directly writable (ADICRz.SHCT = 0001_B or 0010_B).
  – For source shadow addressing (ADICRz.SHCT = 0001_B) the new source address is written to the shadow address register MExSHADR in DMA sub-block x.
  – For destination shadow addressing (ADICRz.SHCT = 0010_B) the new destination address is written to the shadow address register MExSHADR in DMA sub-block x.
  – Direct writes to the shadow address register will result in a bus error.
  – When the current DMA transaction completes and the final write back to DMARAM occurs the MExSHADR value is written to either SADRz or DADRz and the SHADRz is set to 00000000_H.
- Direct Write Mode: the shadow address is directly writable (ADICRz.SHCT = 0101_B or 0110_B) and the shadow address register MExSHADR in DMA sub-block x is directly written.
  – The SHADRz value stored in the DMARAM is updated to the MExSHADR value on the write back.
  – At the start of a new DMA transaction the source address register (MExSADR) or the destination address register (MExDADR) is updated with the SHADRz value. MExADICR.SHCT must be set accordingly.
  – The shadow transfer will be repeated until DMA channel z is reset or until the value in MExSHADR is $0000\ 0000_H$.

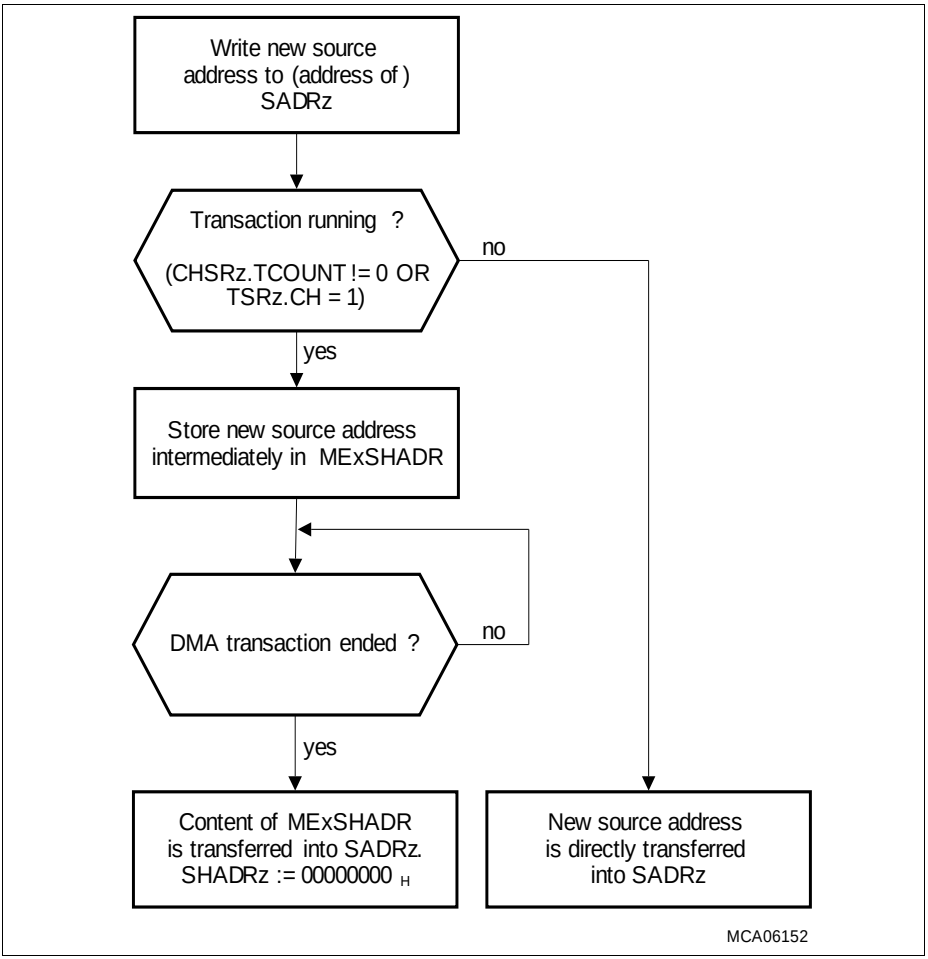


Figure 15-4 Source Address Update (ADICRz.SHCT = 0001_B)

Transfer Count Update

The transfer count of a DMA transaction, stored in bit field CHCFGRz.TREL, can also be programmed if the DMA transaction is running. At the start of a DMA transaction, CHCFGRz.TREL is transferred to bit field MExCHSR.TCOUNT. TCOUNT is updated during the DMA transaction.

No reload of address or counter will be done if MExCHSR.TCOUNT is not equal to 0.

The reprogramming of channel specific values (except for the selected shadow address register) must be avoided while a DMA channel is active as the data transfer maybe corrupted.

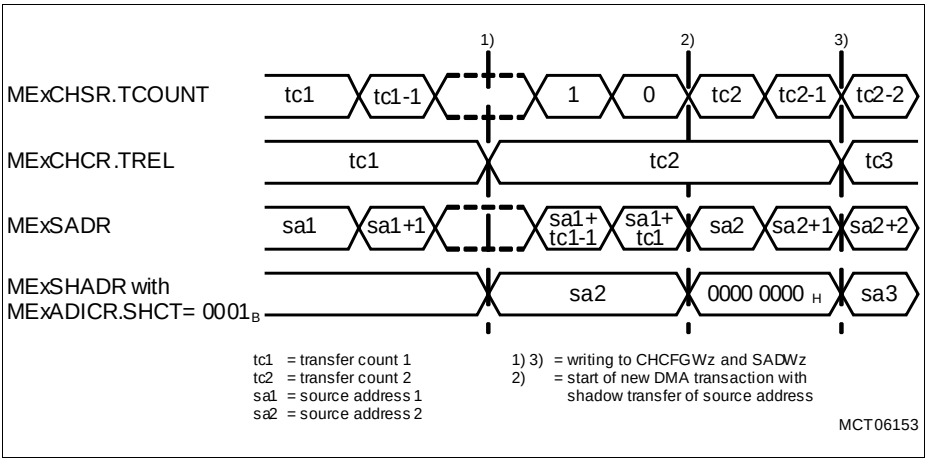


Figure 15-5 Shadow Source Address and Transfer Count Update with MExADICR.SHCT = 0001_B (Shadow Read Only Mode)

Figure 15-5 shows how the contents of the source address register MExSADR and the transfer count MExCHSR.TCOUNT are updated during two DMA transactions with a shadowed source address and transfer count update.

At reference point 2) the DMA transaction 1 is finished and DMA transaction 2 is started. At 1) the DMA channel is reprogrammed with two new parameters for the next DMA transaction: Transfer count tc2 and source address sa2. Source address sa2 is buffered in MExSHADR and transferred to MExSADR when the new DMA transaction is started at 2). At this time, transfer count tc2 is also transferred to MExCHSR.TCOUNT. Note that the shadow address register is only reset by hardware to 0000 0000_H as shown in this example, if the write enable bit is set to 0 (ADICRz.SHCT = 0001_B or 0010_B).

In the event that CHCFGRz.TREL is written while DMA channel z is active:

- A write to CHCFGRz.TREL will update the MExCHCR.TREL value in the DMA sub-block x.
- On write back CHCFGRz.TREL is updated with the latest MExCHCR.TREL value.
- MExCHSR.TCOUNT is updated to the new CHCFGRz.TREL value at the start of the next DMA transaction.

15.4.3.2 DMA Channel Request Control

Figure 15-6 shows the control logic for DMA requests that is implemented for each DMA channel.

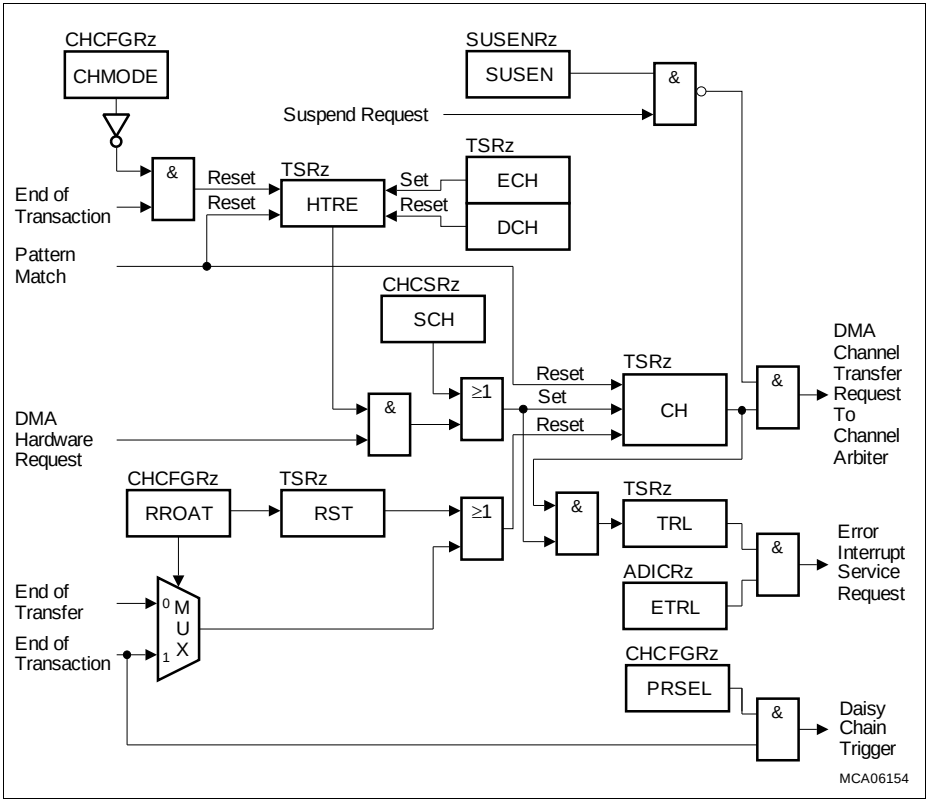


Figure 15-6 Channel Request Control

Two different types of DMA requests are possible:

- Hardware DMA requests
- Software DMA requests

DMA hardware requests (CHz_REQ) are triggered by the Interrupt Control Unit (ICU).

If CHCFGRz.PRSEL = 1 in the current DMA channel z can bypass the ICU and trigger a DMA hardware request in the next lower DMA channel z-1. The latency to service a DMA channel z-1 request is reduced. DMA channel z interrupt service requests are disabled.

Hardware requests are enabled/disabled by status bit TSRz.HTRE. HTRE can be set/reset by software or by hardware in Single Mode at the end of a DMA transaction. A software request can be generated by setting bit CHCSRz.SCH.

Status flag TSRz.CH indicates whether or not a software or hardware generated DMA request for DMA channel z is pending. TSRz.CH is cleared when the DMA transfer starts (RROAT = 0) or at the end of a DMA transaction (RROAT = 1).

If a software or a hardware DMA request is detected for channel z while TSRz.CH is set, a request lost event occurs. This error event indicates that the DMA is already processing a transfer and that another transfer has been requested before the end of the previous one. In this case, bit TSRz.TRL will be set. If ADICRz.ETRL is set then a transfer lost interrupt trigger will be generated when DMA channel z becomes active.

15.4.3.3 DMA Channel Operation Modes

The operation mode of a DMA channel is individually programmable for each DMA channel z. In basic terms, a DMA channel can operate in the following modes:

- Software controlled mode
- Hardware controlled mode, in Single Mode, Continuous Mode or Linked List.

In software-controlled mode, a DMA channel request is generated by setting a control bit. In hardware-controlled mode, a DMA channel request is generated by the ICU servicing an interrupt trigger generated by an on chip peripheral unit.

In hardware-controlled Single Mode, a DMA channel z becomes disabled by hardware after the last DMA transfer of its DMA transaction. In hardware-controlled Continuous Mode, a DMA channel z remains enabled after the last DMA transfer of its DMA transaction.

In linked list mode the DMA channel is loaded with the new transaction control set on completion of the current DMA transaction. The next DMA transaction can optionally be auto started by a software request.

In hardware- and software-controlled mode, a DMA request signal can be configured to trigger a complete DMA transaction or one single transfer.

Software-controlled Modes

In software-controlled mode, one software request starts one complete DMA transaction or one single DMA transfer. Software-controlled modes are selected by writing TSRz.DCH = 1. This forces status flag TSRz.HTRE = 0 (hardware request of DMA channel z is disabled).

The software-controlled mode that initiates one complete DMA transaction to be executed is selected for DMA channel z by the following write operations:

- CHCFGRz.RROAT = 1
- CHCSRz.SCH = 1

Setting CHCSRz.SCH to 1 (this is the software request) causes the DMA transaction of DMA channel z to be started and TSRz.CH to be set. At the start of the DMA transaction, the value of CHCFGRz.TREL is loaded into MExCHSR.TCOUNT (transfer count or tc) and the DMA transfers are executed. After each DMA transfer, TCOUNT becomes decremented and next source and destination addresses are calculated. When TCOUNT reaches the 0, DMA channel z becomes disabled and status flag TSRz.CH is reset. Setting CHCSRz.SCH again starts a new DMA transaction of DMA channel z with the parameters as currently defined in the channel register set.

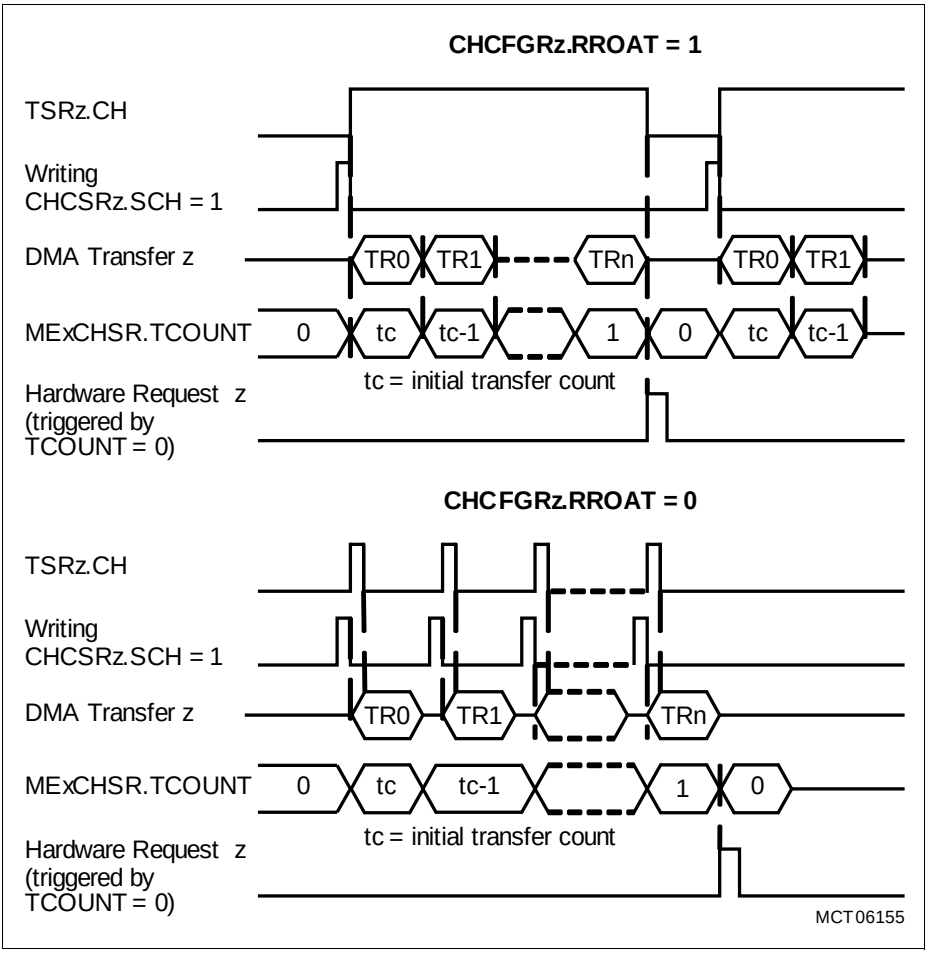


Figure 15-7 Software Controlled Mode Operation

The software-controlled mode that initiates a single DMA transfer to be executed is selected for DMA channel z by the following write operations:

- CHCFGRz.RROAT = 0
- CHCSRz.SCH = 1, repeated for each DMA transfer

When CHCFGRz.RROAT = 0, TSRz.CH is cleared when a DMA transfer starts and a new software request (writing CHCSRz.SCH = 1) must be generated for starting the next DMA transfer.

Hardware-controlled Modes

In hardware-controlled modes, a hardware request signal starts a DMA transaction or a single DMA transfer. There are two hardware-controlled modes available:

- **Single Mode:**
  Hardware requests are disabled by hardware after a DMA transaction
- **Continuous Mode:**
  Hardware requests are not disabled by hardware after a DMA transaction

Hardware-controlled Single Mode

In hardware-controlled Single Modes, one hardware request starts one complete DMA transaction or one single DMA transfer. The hardware-controlled Single Mode that initiates one complete DMA transaction to be executed for DMA channel z is selected by the following operations:

- CHCFGRz.CHMODE = 0
- CHCFGRz.RROAT = 1
- CHCFGRz.PRSEL = 0
- TSRz.ECH = 1

Setting TSRz.ECH to 1 enables hardware transfer requests (TSRz.HTRE = 1) for DMA channel z. When the ICU generates a DMA hardware request CHz_REQ then TSRz.CH is set high. If DMA channel z wins channel arbitration then DMA channel z becomes a move engine active channel. The value of CHCFGRz.TREL is loaded into MExCHSR.TCOUNT and the DMA transaction is started by executing its first DMA transfer. After each DMA transfer, TCOUNT is decremented and next source and destination addresses are calculated. When TCOUNT reaches the 0, DMA channel z becomes disabled and status flags TSRz.CH and TSRz.HTRE are reset. In order to start a new hardware-controlled DMA transaction, hardware requests must be enabled again by setting TSRz.HTRE through TSRz.ECH = 1. The hardware request disable function in Single Mode is typically needed when a reprogramming of the DMA channel register set (addresses, transfer count) is required before the next hardware triggered DMA transaction is started.

The hardware-controlled Single Mode in which each single DMA transfer has to be requested by a hardware request signal is selected as described above, with one difference:

- CHCFGRz.RROAT = 0

In this operation mode, TSRz.CH is cleared when a new DMA transfer starts, and a new hardware request at CHz_REQ must be generated for starting the next DMA transfer.

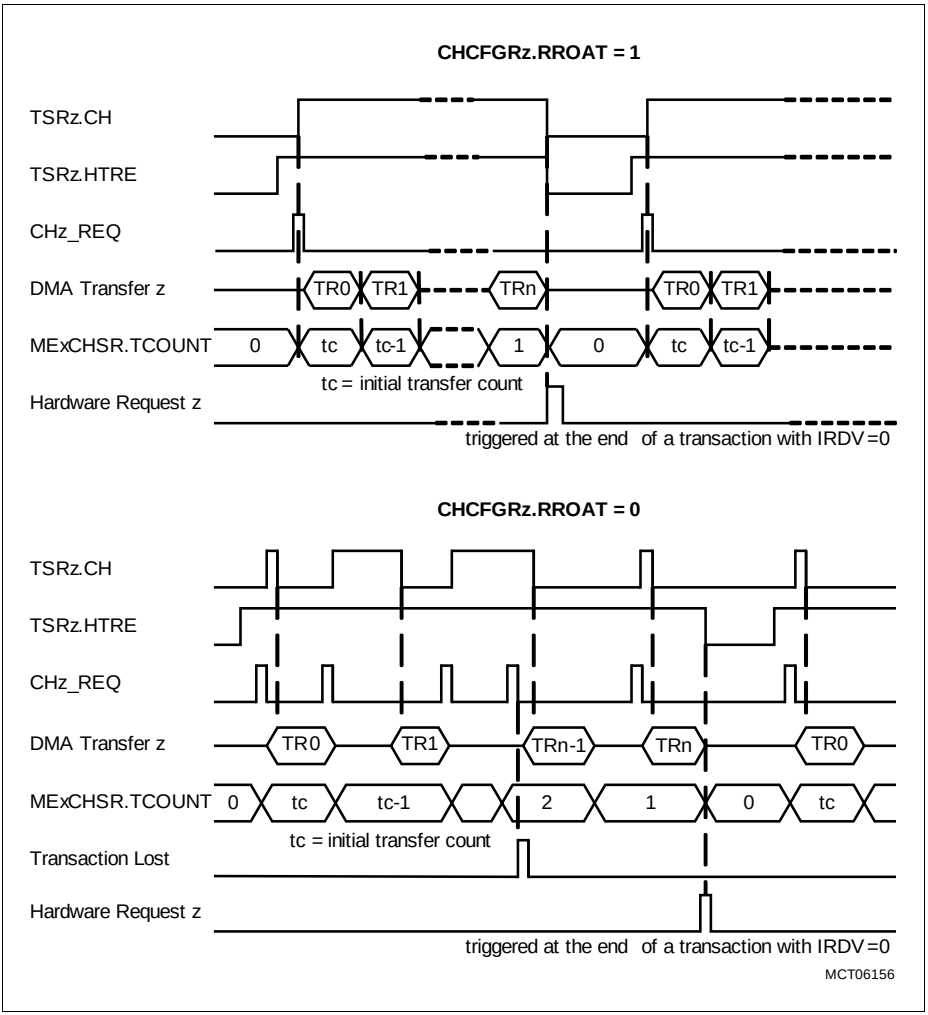


Figure 15-8 Hardware-controlled Single Mode Operation

Hardware-controlled Continuous Mode

In hardware-controlled Continuous Mode (CHCFGRz.CHMODE = 1), the hardware transaction request enable bit TSRz.HTRE is not reset at the end of a DMA transaction. A new transaction of DMA channel z with the parameters actually stored in the transaction control set of DMA channel z is started each time when CHCSRz.TCOUNT = 0 at the end of the DMA transaction. No software re-enable for a hardware request at CHz_REQ is required.

Combined Software/Hardware-controlled Mode

Figure 15-9 shows how software- and hardware-controlled modes can be combined. In the example, the first DMA transfer is triggered by software when setting CHCSRz.SCH. Hardware requests are still disabled. After hardware requests have been enabled by setting TSRz.ECH, subsequent DMA transfers are triggered now by hardware request coming from the CHz_REQ line.

In the example, DMA channel z operates in Single Mode (CHCFGRz.CHMODE = 0). In this mode, TSRz.HTRE becomes reset by hardware when CHCSRz.TCOUNT = 0 at the end of the DMA transaction.

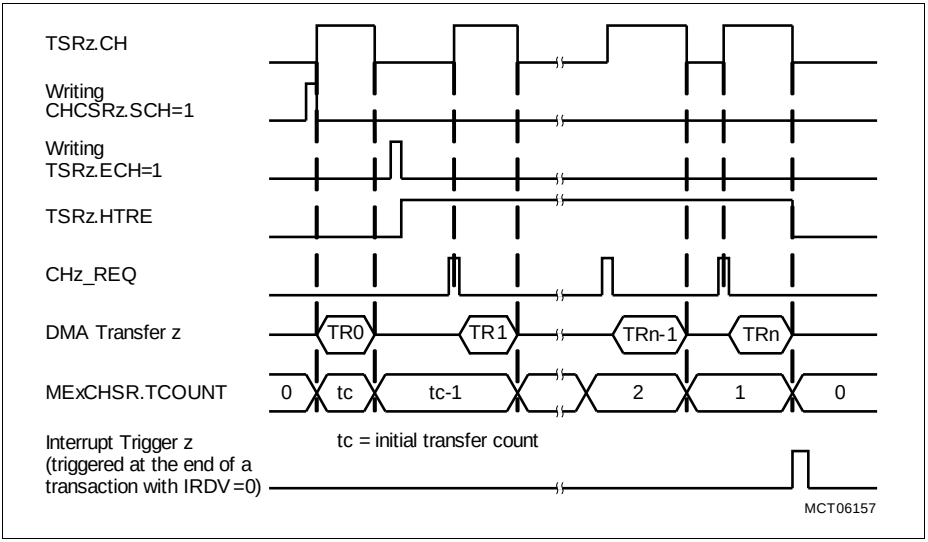


Figure 15-9 Transaction Start by Software, Continuation by Hardware

If a DMA channel is configured to be triggered by parallel hardware and software requests then if the requests collide in the same clock cycle the DMA move will be executed and a Transaction/Transfer Request Lost event will be flagged in the TSRz.TRL bit.

15.4.3.4 DMA Service Requests

Interrupt Requests are prioritized by the Interrupt Router and processed by one of the Service Providers (CPU or DMA). Each DMA peripheral interfaces to a dedicated Interrupt Control Unit (ICU) instantiated in the Interrupt Router (IR).

DMA channels are associated with the Service Request Priority Number bit field programmed in the Service Request Control Register SRC.SRPN. For example:

- DMA Channel 000 equates to SRC.SRPN = 0 programmed in IR.
- DMA Channel 001 equates to SRC.SRPN = 1 programmed in IR.
- DMA Channel 002 equates to SRC.SPRN = 2 programmed in IR.
- DMA Channel 003 equates to SRC.SRPN = 3 programmed in IR.
- DMA Channel 004 equates to SRC.SPRN = 4 programmed in IR.

The routing of a hardware Service Request to a service provider destination is determined by the IR Type of Service Control bit field SRC.TOS. The DMA will acknowledge all Service Requests. If the value programmed in the SRC.SRPN is for an Invalid Channel then the DMA Controller will take no action.

The user must programme valid SRC.SRPN values for a DMA peripheral.

Daisy Chain

DMA Channels can be daisy chained by setting the bit CHCFGRz.PRSEL.

When a higher priority DMA channel completes a DMA transaction then it will initiate a DMA transaction on the next lower priority DMA channel by setting the access pending bit TSRz.CH bit. Daisy chaining is limited to a higher priority DMA channel initiating the next lower priority DMA Channel.

Enabling the daisy chain disables the channel transfer interrupt trigger in the next higher priority channel. In a typical DMA daisy chain application only the lowest priority DMA channel is required to generate a channel transfer interrupt trigger. When the sequence of DMA transactions from the highest to lowest priority DMA channel has completed then the lowest priority DMA channel in the daisy chain will generate a channel transfer interrupt trigger to signal the end of the DMA operation.

Only the Service Request for the highest priority DMA Channel in the daisy chain is initiated by the ICU. The lower priority Service Requests bypass the ICU in order to improve interrupt latency.

15.4.3.5 Channel Reset Operation

A DMA transaction of DMA channel z can be stopped (channel is reset) by setting bit TSRz.RST. If a read or write on chip bus access of DMA channel z is executed at the time when TSRz.RST is set then the On Chip Bus access is finished normally. This behavior guarantees data consistency.

When a channel reset is applied to an active DMA channel, the on-going DMA transfer executing in the move engine will complete before the DMA channel transitions to the reset state. A pending DMA channel must become active and service its pending request before making a transition to the channel reset state.

DMA Channel Reset State

On completion of a DMA channel reset:

- Bits TSRz.HTRE, TSRz.CH, TSRz.TRL, CHCSRz.ICH, CHCSRz.IPM, CHCSRz.WRPD, CHCSRz.WRPS, CHCSRz.LXO, and bit field CHCSRz.TCOUNT are reset.
- If a circular buffer ADICRz.SCBE and/or ADICRz.DCBE is enabled then the source and/or destination address register will be set to the wrap boundary else the address registers are cleared. SHADRz will be cleared.
- All automatic functions are stopped for channel z.

Resetting a DMA Channel

A user program must execute the following steps to reset a DMA channel:

1. If hardware requests are enabled for the DMA channel z, disable the DMA channel z hardware requests by setting TSRz.DCH = 1.
2. Writing a 1 to TSRz.RST.
3. Waiting (polling) until TSRz.RST = 0 to indicate the reset has completed.

During a DMA channel reset operation, software requests must not be initiated by setting CHCSRz.SCH = 1.

Restarting a DMA Channel

A user program must execute the following steps for restarting a DMA channel after it was reset:

1. Optionally (re-)configuring the address and other channel registers.
2. Restarting the DMA channel z by setting TSRz.ECH = 1 for hardware requests or CHCSRz.SCH = 1 for software requests.

15.4.3.6 Channel Halt Operation

A DMA channel can be halted during a DMA transaction and the state frozen to allow a background RAM test to be run over a destination memory in order to detect stuck bits and distinguish between static errors and transient errors. On completion of the RAM test the DMA channel can be re-started and the DMA transaction completed.

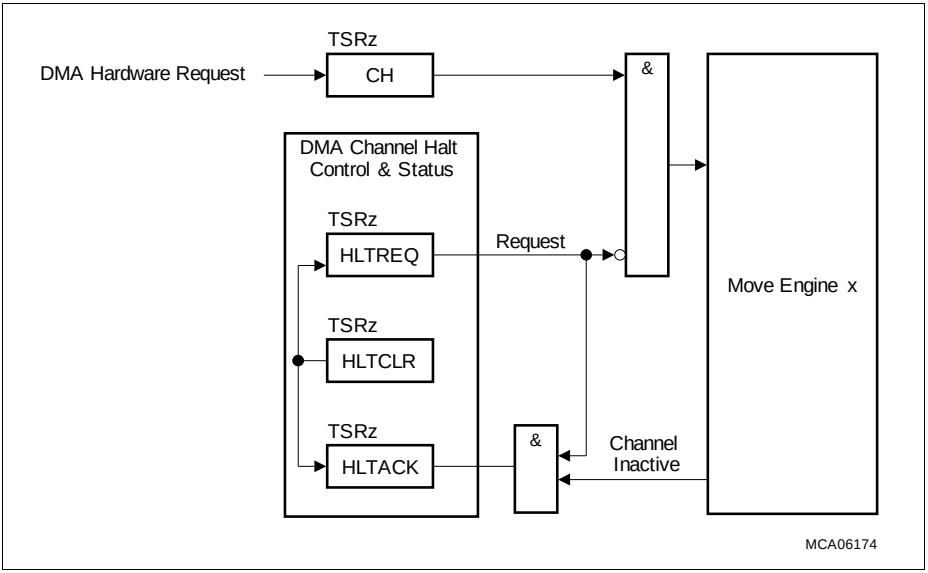


Figure 15-10 DMA Channel Halt Control

The halt control utilizes a set/clear mechanism to request the channel transitions to and from the HALT state on completion of the current DMA transfer. Only writing a logic ‘1’ to set or clear a halt request to a DMA channel has an effect and the status of other channels can be ignored.

The DMA channel halt operation is shown in **Figure 15-11**.

Entering DMA Channel Halt

Channel Halt Mode of DMA channel z is entered if the halt request bit TSRz.HLTREQ is set by software. The DMA channel enters the halt state mode automatically after the current DMA transfer is completed. The DMA channel z halt state is signalled by the halt acknowledge status flag TSRz.HLTACK.

When the DMA channel is in the halt state the transaction control set can be modified. These modifications are taken into account for further DMA transactions or DMA transfers of the related DMA channel after Halt Mode has been left again.

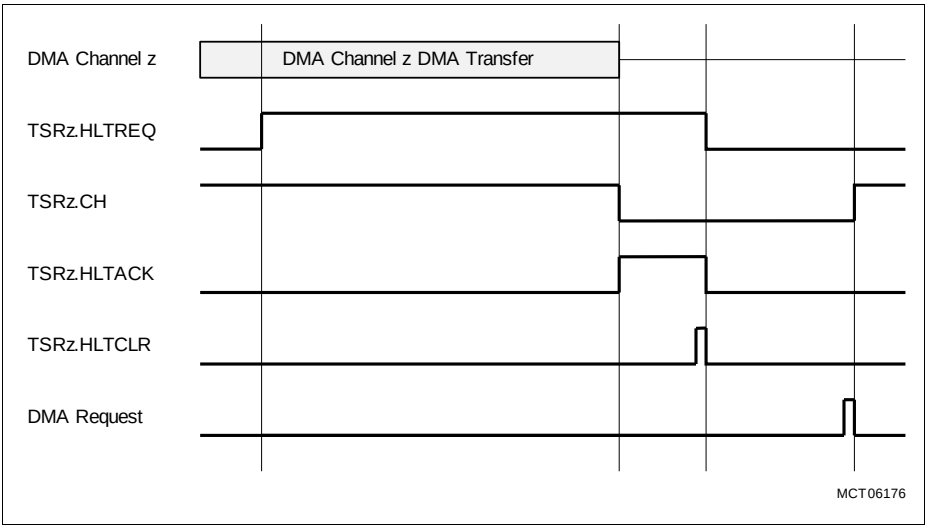


Figure 15-11 DMA Channel Halt Operation

Exiting Channel Halt Mode

DMA channel z is released from the halt state by software setting the halt clear bit TSRz.HLTCLR. Normal DMA operation is resumed.

DMA Hardware Request during Channel Halt Mode

If DMA channel z is in the halt state (TSRz.HLTACK = 1_B) and the hardware transaction request enable bit TSRz.HTRE is set then the DMA channel z will respond to DMA hardware request service requests as follows:

- Idle channel (TSRz.CH = 0_B): the access pending bit will be set and the DMA hardware request will be serviced when DMA channel z exits halt mode.
- Active channel (TSRz.CH = 1_B): the TSRz.TRL bit is set as the transaction will be lost. If the enable transaction request lost interrupt bit ADICRz.ETRL is set then an interrupt will be generated for a transaction request lost event.

15.4.3.7 Transfer Count and Move Count

The move count determines the number of moves (consisting of one read and one write each) to be done in each DMA transfer. It allows the user to indicate to the DMA the number of moves to be done after one request. The number of DMA moves per DMA transfer is selected by the block mode settings (CHCFGRz.BLKM).

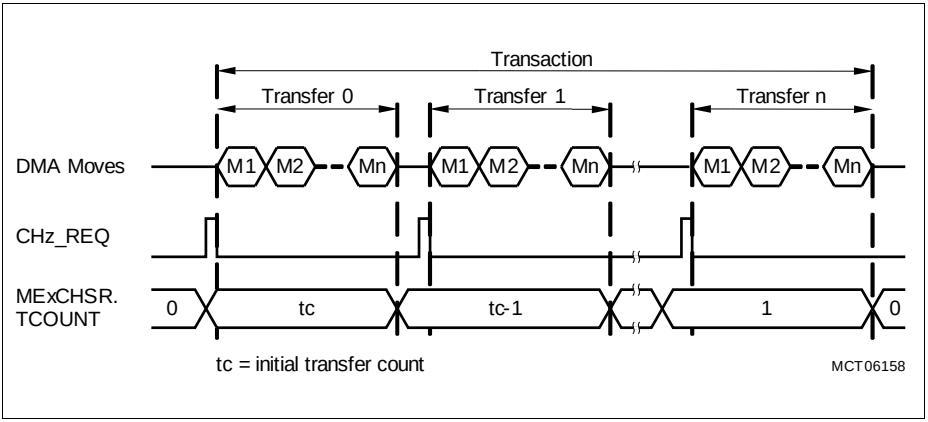


Figure 15-12 Transfer and Move Count

After a DMA move, the next source and destination addresses are calculated. Source and destination addresses are calculated independently of each other. The following address calculation parameters can be selected:

- The address offset, which is a multiple of the selected data width
- The offset direction: addition, subtraction, or none (unchanged address)

Control bits in address and interrupt control word ADICRz determine how the addresses are incremented/decremented. Further, the data width as defined in CHCFGRz.CHDW is taken into account for the address calculation.

Figure 15-13 and **Figure 15-14** show two examples of address calculation. In both examples, a data width of 16-bit (CHCFGRz.CHDW = 001_B) is assumed.

Programmable Address Modification - Example 1

In **Figure 15-13**, 16-bit half-words are transferred from a source memory with an incrementing source address offset of 10_H to a destination memory with decrementing destination addresses offset of 08_H.

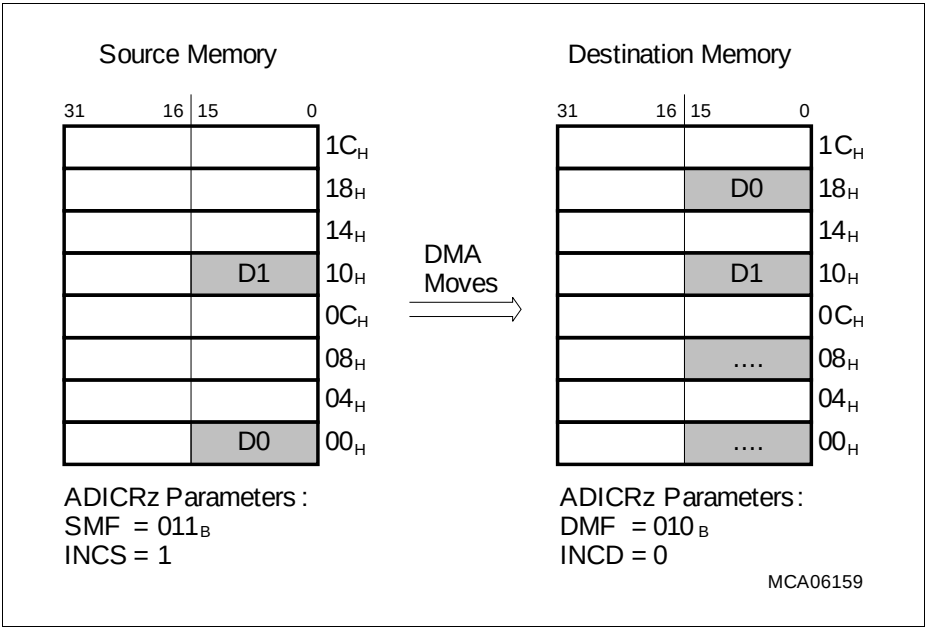


Figure 15-13 Programmable Address Modification - Example 1

Programmable Address Modification - Example 2

In **Figure 15-14**, 16-bit half-words are transferred from a source memory with an incrementing source address offset of 02_H to a destination memory with incrementing destination addresses offset of 04_H.

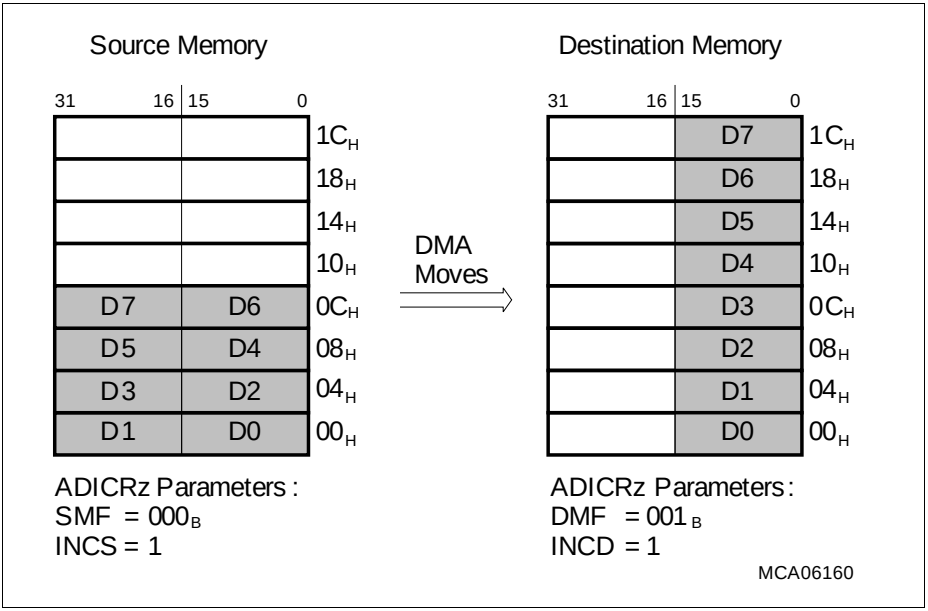


Figure 15-14 Programmable Address Modification - Example 2

15.4.3.8 Circular Buffer

Destination and source address can be configured to build a circular buffer separately for source and destination data. Within this circular buffer, addresses are updated as defined in **Figure 15-13** and **Figure 15-14** with a wrap-around at the buffer limits. The circular buffer length is determined by bit fields ADICRz.CBLS (for the source buffer) and ADICRz.CBLD (for the destination buffer). These 4-bit wide bit fields determine which bits of the 32-bit address remain unchanged at an address update. Possible buffer sizes of the circular buffers can be 2^{CBLS} or 2^{CBLD} bytes (= 1, 2, 4, 8, 16, … up to 64k bytes).

Source and destination addresses are updated (incremented or decremented) during a DMA move, all upper bits [31:CBLS] of source address and [31:CBLD] of destination address are frozen and remain unchanged, even if a wrap-around from the lower address bits [CBLS-1:0] or [CBLD-1:0] occurred. This address-freezing mechanism always causes the circular buffers to be aligned to a multiple integer value of its size.

If the circular buffer size is less or equal than the selected address offset (see **Figure 15-13**), the same circular buffer address will always be accessed.

The source and destination circular buffers are enabled by setting the circular buffer enable bits ADICRz.SCBE and ADICRz.DCBE respectively.

15.4.3.9 Address Counter

If ADICRz.SCBE = 0 then the source circular buffer is not enabled and the address will increment/decrement determined by ADICRz.INCS across the entire 32-bit address field. The address offset is determined by CHCFGRz.CHDW. The source address will wrap around on the 32-bit address boundary:

- If incrementing will wrap from $FFFFFFFF_H$ to 00000000_H.
- If decrementing will wrap from 00000000_H to $FFFFFFFF_H$.

15.4.3.10 Flow Control

A timestamp can optionally be attached at the end a DMA transaction to record the occurrence of the event. The timestamp is the time at which the event was recorded by the DMA controller and not the real time.

Timestamp Generation

The timestamp is generated from a 32-bit binary upwards synchronous counter clocked by the SPB clock divided by 8 as shown in **Figure 15-15**. The counting starts automatically after an application reset. It is not possible to affect the counter during normal operation.

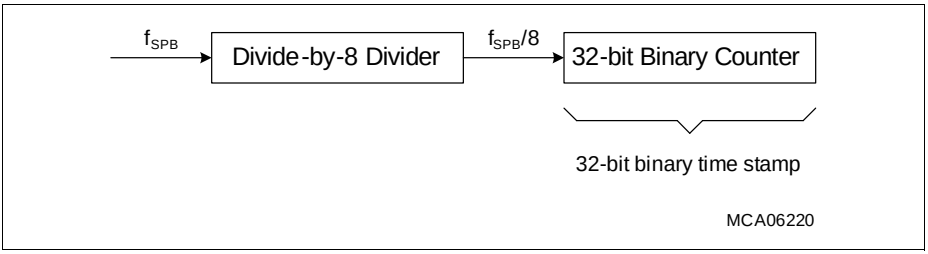


Figure 15-15 Timestamp Generation

The current 32-bit count value can be read through the DMA_TIME register.

Timestamp Appendage

The appendage of a timestamp to the destination data at the end of a DMA transaction is controlled by the ADICRz.STAMP bit. If enabled then a 32-bit timestamp will be appended by a DMA write move to the next word aligned address in the destination data

sequence. If the address control increment of destination address bit ADICRz.INCD = 1_B then the timestamp will be written at the next higher word aligned destination address and if ADICRz.INCD = 0_B then the timestamp will be written at the next lower word aligned destination address.

If a DMA channel is configured to support circular buffer operation then the timestamp must be appended at the next word aligned address above the circular buffer (ADICRz.INCD = 1_B) or below the circular buffer (ADICRz.INCD = 0_B). The appendage of the timestamp must not overwrite or change the destination address of circular buffer DMA move data.

Appendage of Timestamp - Example 1

In **Figure 15-16**, 16-bit half-words are transferred from a source memory with an incrementing source address offset of 10_H to a destination memory with decrementing destination addresses offset of 08_H.

The destination address is incrementing (ADICRz.INCD = 1_B) and the timestamp is appended above the destination data.

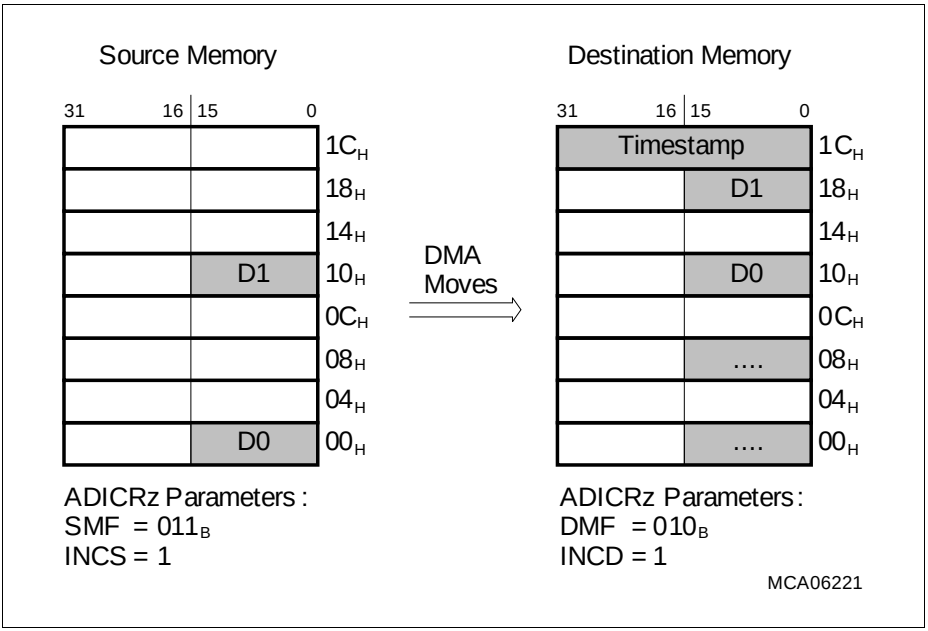


Figure 15-16 Appendage of Timestamp - Example 1 (ADICRz.INCD = 1)

Appendage of Timestamp - Example 2

In **Figure 15-17**, 16-bit half-words are transferred from a source memory with an incrementing source address offset of 02_H to a destination memory with decrementing destination addresses offset of 04_H.

The destination address is decrementing (ADICRz.INCD = 0) and the timestamp is appended below the destination data.

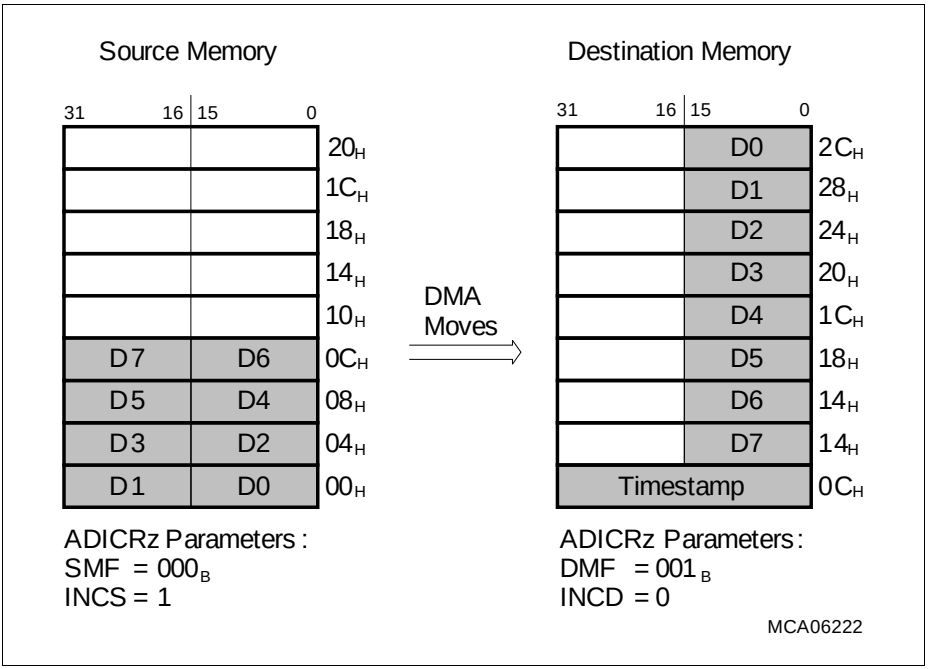


Figure 15-17 Appendage of Timestamp - Example 2 (ADICRz.INCD = 0)

Flow Control

The appendage of timestamps to the end of DMA transactions support flow control. The relative count value of the timestamp can be used to determine the sequence of DMA writes to different destination addresses.

15.4.3.11 Double Buffering Operation

The DMA supports double buffering **Figure 15-18** shows an example of double destination buffering. DMA read moves transfer a continuous data stream from a peripheral to the DMA controller and DMA write moves transfer the read data from the DMA controller to one of two destination buffers stored in memory. The data stream can be re-directed via a software switch. The dormant data buffer can be frozen and be available for cyclic software tasks while the other data buffer continues to be filled.

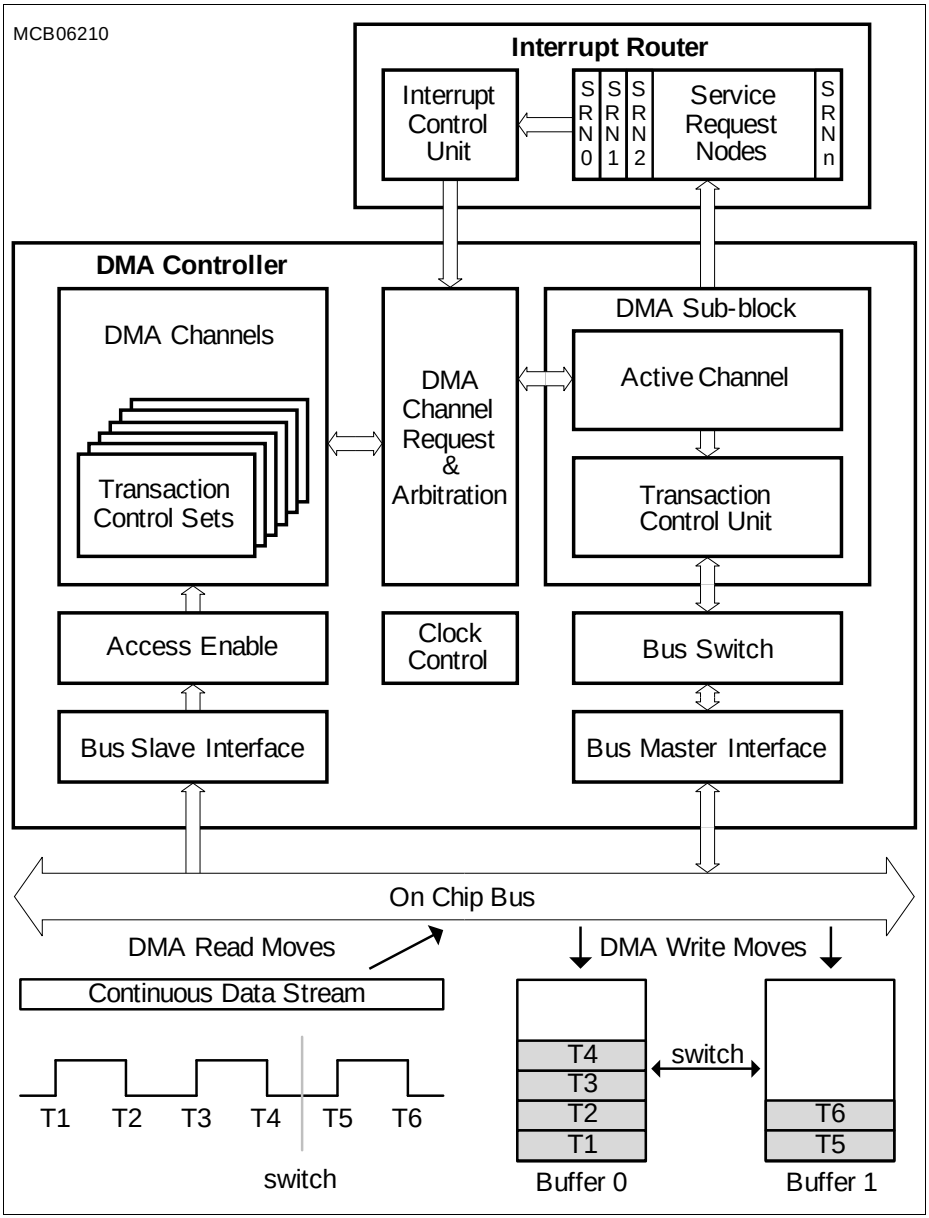


Figure 15-18 DMA Double Buffering

Double Source Addressing

A DMA Channel can be configured to support double source addresses (ADICRz.SHCT[3:1] = 100_B).

Data is streamed by the DMA from one of two data buffers (buffer 0 and buffer 1) stored in memory to a continuous output data stream. Double buffer source addresses:

- The source address SADRz is used to address buffer 0.
- The shadow address SHADRz is used as a source address to address buffer 1.

The size of the two source double buffers is equal and determined by the following DMA transaction control parameters:

- The channel data width (CHCFGRz.CHDW).
- The source circular buffer enable/disable control (ADICRz.SCBE).
- The source address modification factor (ADICRz.SMF).
- The increment of the source address (ADICRz.INCS).
- The block mode value (CHCFGRz.BLKM).
- The transfer reload value (CHCFGRz.TREL).

The size of one source buffer is equal to the size of one DMA transaction.

Double Destination Addressing

A DMA Channel can be configured to support double destination addresses (ADICRz.SHCT[3:1] = 101_B).

A continuous input data stream is directed by the DMA to one of two data buffers (buffer 0 and buffer 1) stored in memory. Double buffer destination addresses:

- The destination address DADRz is used to address buffer 0.
- The shadow address SHADRz is used as a destination address to address buffer 1.

The size of the two destination double buffers is equal and determined by the following DMA transaction parameters:

- The channel data width (CHCFGRz.CHDW).
- The destination circular buffer enable/disable control (ADICRz.DCBE).
- The destination address modification factor (ADICRz.DMF).
- The increment of the destination address (ADICRz.INCD).
- The block mode value (CHCFGRz.BLKM).
- The transfer reload value (CHCFGRz.TREL).

The size of one destination buffer is equal to the size of one DMA transaction.

Active Buffer

During DMA double buffering the CHCSRz.BUFFER status bit is used to indicate which buffer is active.

- MExCHSR.BUFFER = 0_B: buffer 0 is read or filled by the DMA.

- MExCHSR.BUFFER = 1_B: buffer 1 is read or filled by the DMA.

Buffer Switching

The re-direction of the data stream from one buffer to the other can be controlled by software and additionally automatically by hardware. Automatic hardware switching is selected by setting ADICRz.SHCT[0] = 1_B. The different DMA double buffering modes are shown in **Table 15-1**.

Table 15-1 DMA Double Buffering Modes

SHCT	Description
1000_B	**DMA Double Source Buffering** **Software Switch Only** Switching from the current source buffer to the other source buffer is controlled by the software switch CHCSRz.SWB
1001_B	**DMA Double Source Buffering** **Automatic Hardware and Software Switch** DMA controller automatically switches source buffers when current source buffer is emptied. Software source buffer switching controlled by CHCSRz.SWB
1010_B	**DMA Double Destination Buffering** **Software Switch Only** Switching from the current destination buffer to the other destination buffer is controlled by the software switch CHCSRz.SWB
1011_B	**DMA Double Destination Buffering** **Automatic Hardware and Software Switch** DMA controller automatically switches destination buffers when current destination buffer is full. Software destination buffer switching controlled by CHCSRz.SWB

Software Buffer Switch

Before a DMA transaction completes and the active buffer is full software can re-direct the data stream from the active buffer to the other buffer by setting CHCSRz.SWB. If a move engine is actively servicing a DMA channel z configured for double buffering then the current DMA transfer completes before the buffer switch is made.

On completion of the DMA transfer the DMA controller will:

- Automatically re-load MExCHSR.TCOUNT with CHCFGRz.TREL
- Switch the address pointer between the source/destination address and the shadow
  - Buffer 0 address pointer (source or destination address).
  - Buffer 1 address pointer (shadow address).

The address control factors remain the same. The next DMA transaction controlling the filling or reading of the new buffer will start when a DMA request is received.

During a software buffer switch no DMA requests are lost by the DMA controller and there is no loss, duplication or split of data across the two buffers.

Buffer Empty or Full

If the current buffer is emptied (in the case of DMA double source buffering) or filled (in the case of DMA double destination buffering) before a software buffer switch is received then the DMA channel will stop and no more DMA transfers are made. The transaction control set is frozen. The transfer count status (MExCHSR.TCOUNT and CHCSRz.TCOUNT) when the DMA channel is stopped is equal to 0000_H.

The signalling of a buffer empty or full event can be generated by using the DMA channel traffic interrupt service request. The interrupt control ADICRz.INTCT is set to 10_B and the interrupt raise detect value ADICRz.IRDV set to 0000_B. When the transfer count equals zero then the DMA channel will raise a DMA channel interrupt service request.

The DMA channel interrupt handler can interrogate the DMA channel transaction control set to identify which empty or full buffer generated the interrupt. Software can reset the channel ready to restart DMA double buffering operations.

Fast Data Rate

If the data rate is faster than the DMA controller is able to transfer the data to one of the buffers then a transaction request lost event will occur.

If the DMA controller is servicing a DMA access pending request (TSRz.CH = 1_B) and a DMA hardware request is detected then the transaction request lost bit TSRz.TRL will be set. If the enable transaction request lost bit is set (ADICRz.ETRL = 1_B) then a DMA error interrupt service request will be generated.

The DMA error interrupt handler can interrogate the DMA channels and identify the source of the error. Software can reset the DMA channel and restart DMA double buffer operations.

Automatic Hardware Buffer Switch

The DMA controller will automatically switch from the active buffer to the other buffer when the DMA transaction reading or filling the active buffer completes. On switching buffers the MExCHSR.FROZEN bit is set to indicate that one buffer is frozen and available for cyclic software tasks. The interrupt control bits ADICRz.INTCT and interrupt raise detect value MExADICR.IRDV can be used to generate a DMA channel interrupt service request.

On completion of the DMA transaction when MExCHSR.TCOUNT equals 0000_H the DMA controller will:

- Automatically re-load MExCHSR.TCOUNT with CHCFGRz.TREL
- Switch the address pointer between the source/destination address and the shadow address:
  – Buffer 0 address pointer (source or destination address).
  – Buffer 1 address pointer (shadow address).

The address control factors remain the same. The next DMA transaction controlling the filling or reading of the new buffer will start when a DMA request is received.

During an automatic hardware buffer switch no DMA requests are lost by the DMA controller and there is no loss, duplication or split of data across the two buffers.

Frozen Buffer

When the DMA controller switches buffer the frozen bit MExCHSR.FROZEN is set. The frozen buffer is then available for a cyclic software task. On completion of the software task the status bit MExCHSR.FROZEN is cleared by software and the buffer address pointer is re-programmed to the start address ready for the next DMA transaction that reads/fills the buffer.

Transaction Request Lost

If MExCHSR.FROZEN is equal to 1_B then the automatic hardware switch buffer will not occur and the DMA controller will stop. If a DMA hardware request is detected before MExCHSR.FROZEN is cleared then the transaction request lost bit TSRz.TRL is set and a DMA error interrupt service request will be generated if ADICRz.ETRL is 1_B.

DMA Double Buffer Operation

A typical DMA double buffer application is shown in **Figure 15-19**:

- A DMA read move loads a continuous stream of 32-bit data words from an SPB connected peripheral into the DMA controller.
- A DMA write move stores the data from the DMA controller into one of two destination data buffers.

Double destination buffering with software buffer switch only is selected by using the shadow operation control bits (ADICRz.SHCT = 1010_B) as shown in **Table 15-1**. The shadow mechanism is not available.

Transaction Control Set

The DMA active channel points at the two buffers as follows:

- The destination address register MExDADR.DADR is the address pointer to buffer 0.
- The shadow address register MExSHADR.SHADR is the address pointer to buffer 1.

The transfer count (CHCFGRz.TREL), block mode (CHCFGRz.BLKM) and address modification factors (CHCFGRz.CHDW, ADICRz.DCBE, ADICRz.DMF and ADICRz.INCD) are the same for the two buffers.

The buffer depth of the two data buffers is identical and is equal to one DMA transaction. The size of the buffer depth is a multiple of the address modification factors, the DMA transfer block mode and the DMA transaction transfer count.

DMA Operation

The DMA channel z configured for double destination buffering is programmed to execute DMA transfers of 2 X 32-bit words from a source peripheral to one of two destination buffers.

The DMA channel receives a DMA request and on winning arbitration executes a DMA transfer. During the second DMA transfer, the DMA channel z receives a swap buffer command. On completion of the current DMA transfer the frozen buffer bit MExCHSR.FROZEN is set.

At the start of the next DMA transfer the incoming data stream is re-directed to buffer 1. No data is lost, duplicated or split between the buffers. MExCHSR.TCOUNT is reloaded with CHCFGRz.TREL and executes DMA moves to the buffer 1 starting at the address stored in MExSHADR. The status bit MExCHSR.BUFFER switches state to indicate that the DMA controller is filling buffer 1. The cyclic software task operates on buffer 0 and on completion clears MExCHSR.FROZEN and re-programmes the destination address register to the buffer 0 start address. Buffer 0 is now available to receive data and the software will issue a switch buffer instruction and in turn buffer 1 will become available for a cyclic software task.

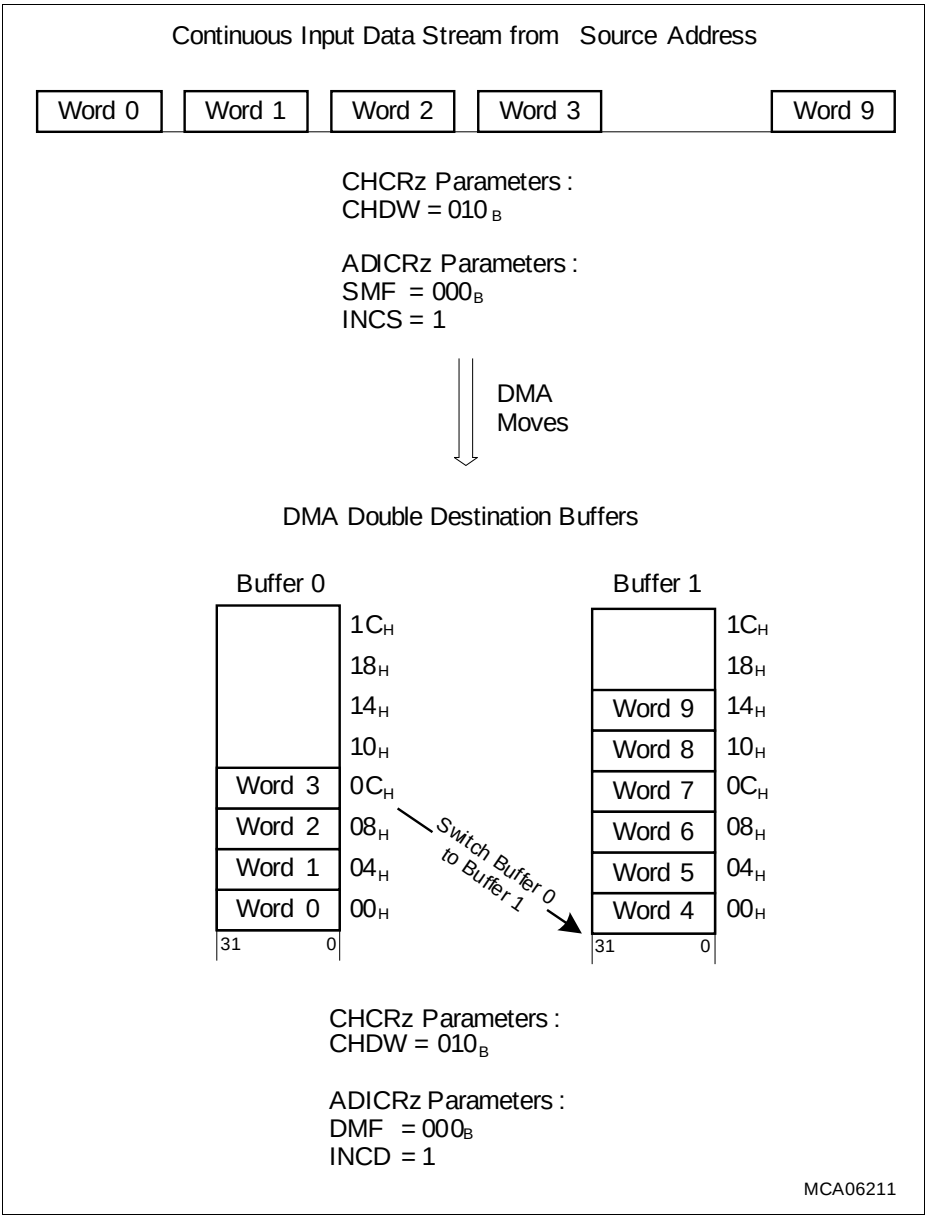


Figure 15-19 DMA Double Destination Buffering

15.4.3.12 Linked Lists

Linked lists are an extension of the DMA channel functionality. Linked lists consist of a series of DMA transactions executed by the same DMA channel z. Each DMA transaction has a unique transaction control set. The source and destination areas do not have to exist in contiguous areas of memory. The new DMA transaction can issue a DMA software request and auto start the DMA transaction bypassing the Interrupt Router and so reducing the cumulative latency over a number of DMA transactions.

In a linked list, the current DMA channel transaction control set defines the shadow register as an address pointer to read the next 32-byte transaction control set from anywhere in memory and overwrite the current transaction control set in the DMARAM.

The following types of linked lists are supported:

- DMA Linked List (see **Section 15.4.3.13**)
- Accumulated Linked List (see **Section 15.4.3.14**)
- Safe Linked List (see **Section 15.4.3.15**)
- Conditional Linked List (see **Section 15.4.3.16**)

Linked lists support greater flexibility in the use of the DMA controller.

Circular Operation of Linked List

Linked lists may be configured for circular operation when the same series of DMA transactions in the linked list are endlessly repeated.

Pattern Detection During a Linked List

If a DMA transaction in a DMA, Accumulated or Safe Linked List is configured for pattern detection then a pattern match (see **Section 15.4.7**) will terminate linked list operation. If the pattern match occurs during the last DMA move of the DMA transaction then the current DMA move status is preserved and the next transaction control set is not loaded. Software must reconfigure the DMA channel.

Software Termination of a Linked List

If the linked list is terminated by software then the current DMA transaction is completed and the linked list is disabled.

15.4.3.13 DMA Linked List

A DMA channel z supports a DMA linked list consisting of 32-Byte transaction control sets by using the shadow operation control bits (ADICRz.SHCT = 1100_B) as shown in **Table 15-15**.

If DMA channel z is configured to support a DMA linked list then when the DMA move engine x completes the servicing of a DMA request from DMA channel z it will read the next transaction control set from memory and overwrite the DMARAM channel z with the new transaction control set. The current DMA transaction uses the shadow address as an address pointer to the next transaction control set. The address pointer to the next transaction control set must be mapped to a 32-byte aligned address in either internal or external memory and configured as shown in **Figure 15-20**. A BTR4 access can be used to read the new transaction control set. The DMA controller will start reading the new transaction control set from the RDCRCRz word upwards.The last word to be read is CHCSRz.

If CHCSRz.SCH is set to 1 then the new transaction control set will auto start and initiate a new transaction request for DMA channel z. The access pending bit TSRz.CH will be set. The DMA controller will arbitrate against other pending requests on completion of the current DMA transfer. When DMA channel z wins arbitration the new DMA transaction will be serviced. There is no limit to the number of DMA transactions in a DMA linked list.

The DMA transactions in the linked list can be started by hardware or software requests. Auto start of a new DMA channel z transaction control set is limited to software requests.

Last DMA Transaction in DMA Linked List

The DMA traffic management interrupt service requests may be used to signal the completion of the last DMA transaction in the DMA linked list. The last DMA transaction will load a new transaction control set. The auto start is not to be set.

Loading Non Linked List Transaction Control Set

If a DMA linked list loads a non linked list transaction control set then the DMA channel exits linked list mode. The non linked list transaction control set will not auto start.

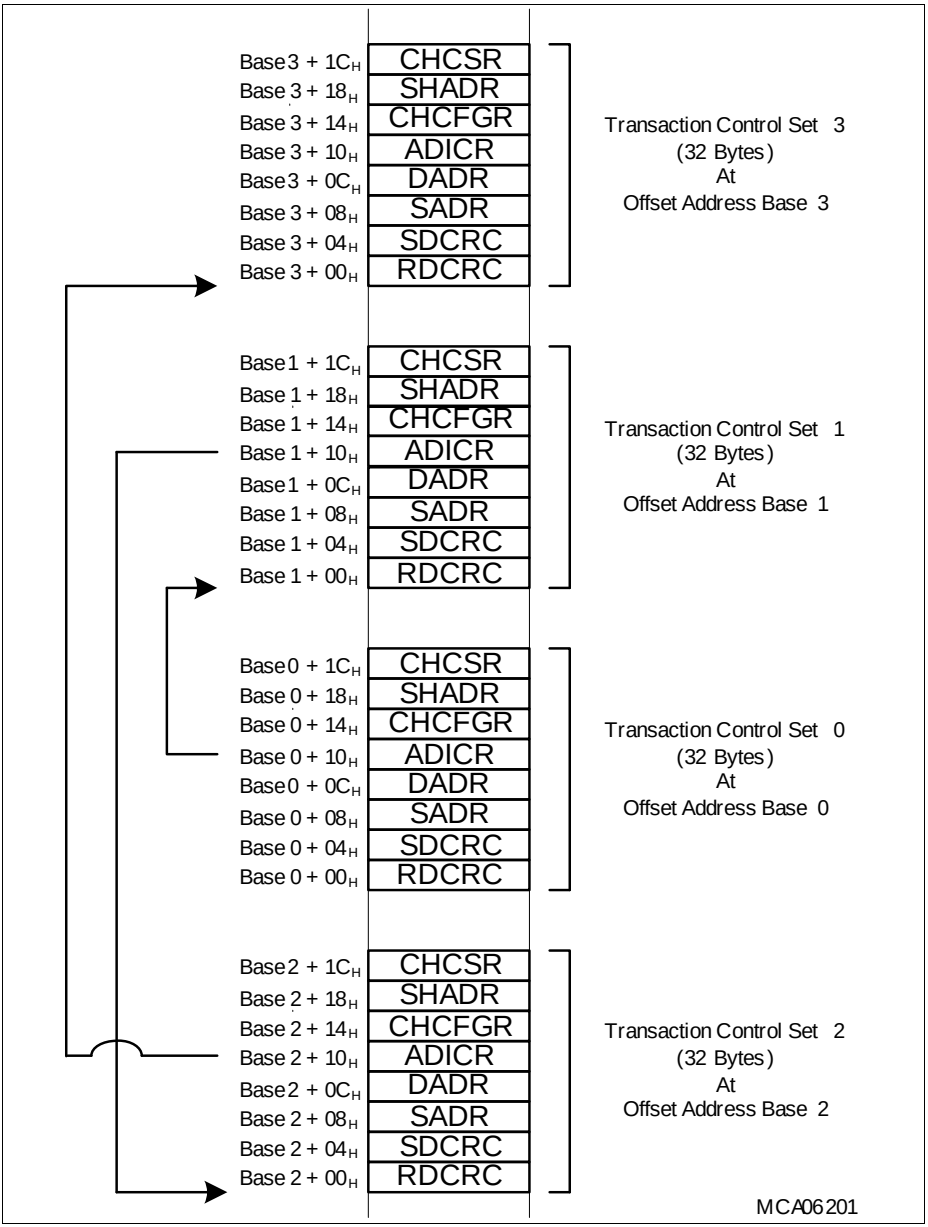


Figure 15-20 DMA Linked List

15.4.3.14 Accumulated Linked List

A DMA channel z supports an accumulated linked list by using the shadow operation control bits (ADICRz.SHCT = 1101_B) as shown in **Table 15-15**. The accumulated linked list is a variant of the DMA linked list and has an identical footprint in memory (size and structure). The SDCRCRz and RDCRCRz words are not overwritten when the new transaction control set is written into the DMARAM channel z and are accumulated across DMA transactions.

SDCRC & RDCRC Checksums

The SDCRC and RDCRC checksums are not overwritten when the new transaction control set is loaded into the DMA channel. On completion of the linked list operation (on completion of the last DMA transaction) the SDCRC and RDCRC checksums are the values calculated across all DMA transactions.

15.4.3.15 Safe Linked List

A Safe linked list provides protection against software errors. If the shadow address in the linked list is incorrectly written then a link address pointer could point to any random area of memory and load a new transaction control set and execute it. A DMA channel z supports a safe linked list by using the shadow operation control bits (ADICRz.SHCT = 1110_B) as shown in **Table 15-15**.

Proceeding from one DMA transaction to the next DMA transaction in the linked list sequence is dependent on the current SDCRC checksum matching the expected SDCRC checksum stored in the next transaction control set.

SDCRC Checksum

The user is required to load the SDCRC word with an expected CRC checksum value. When the safe link list reads the new transaction control set from memory the SDCRC checksum generated by the running DMA transaction is compared against the new expected SDCRC checksum. If the checksums match then the DMA controller proceeds with the execution of the new transaction control set. And if not then a DMA error service interrupt request is triggered, error flag ERRxSR.SLLER set and the DMA stops the execution of the linked list. Assuming all the SDCRC checksum values match at the end of the linked list then the checksum is the value calculated across all DMA transactions.

RDCRC Checksum

The RDCRC checksum word is not overwritten when the new transaction control set is loaded into the DMA channel. The RDCRC checksum value at the end of the linked list is the accumulated checksum across the read data for all DMA transactions.

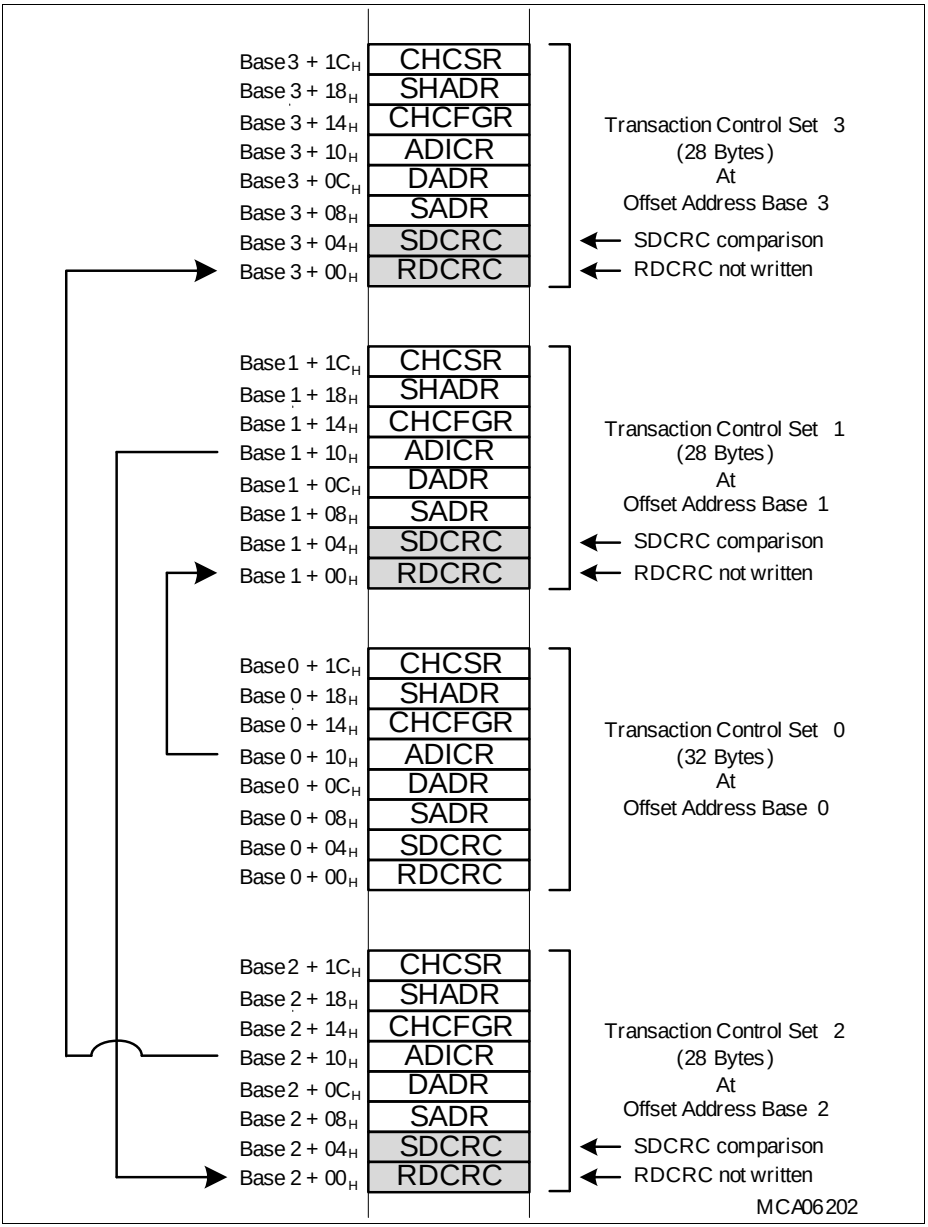


Figure 15-21 Safe Linked List

15.4.3.16 Conditional Linked List

A special use of linked lists is the Conditional Linked Lists (CLL) as shown in **Figure 15-22**. Selection of the address pointer is determined by a conditional state.

A DMA channel supports a Conditional Linked List by using the shadow operation control bits (ADICRz.SHCT = 1111_B) as shown in **Table 15-15**. DMA usage of CLL is limited to 8-bit DMA read moves (CHCFGRz.CHDW = 000_B). CLL does not support the appendage of timestamps (ADICRz.STAMP = 0_B).

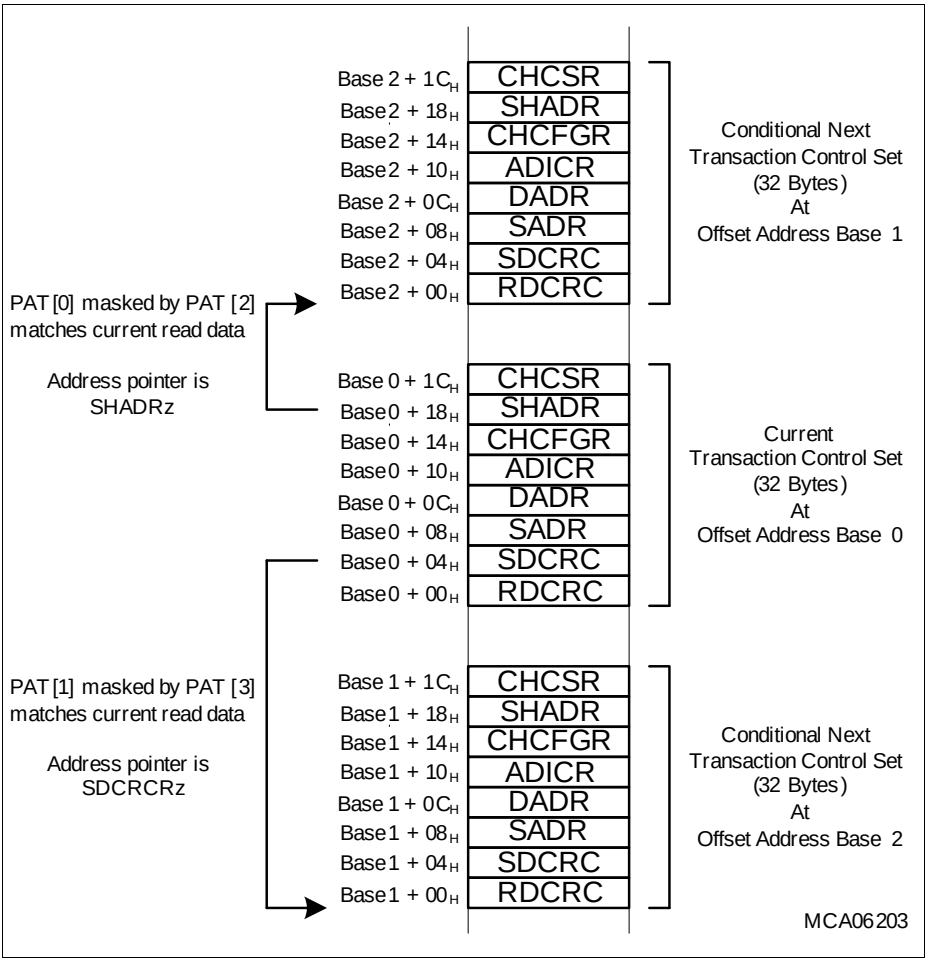


Figure 15-22 Conditional Linked List

In addition to using the shadow address as an address pointer CLL utilizes the source and destination address CRC checksum register SDCRCRz as a second address pointer.

CLL Pattern Detection

The DMA channel must be programmed to select one of the pattern read registers:

- PRR0 is selected if CHCFGRz.PATSEL = 000_B
- PRR1 is selected if CHCFGRz.PATSEL = 100_B

Note: If a DMA channel is configured for CLL then CHCFGRz.PATSEL[1:0] = 00_B

The configuration of the pattern detection logic to support conditional linked lists is shown in **Figure 15-23**.

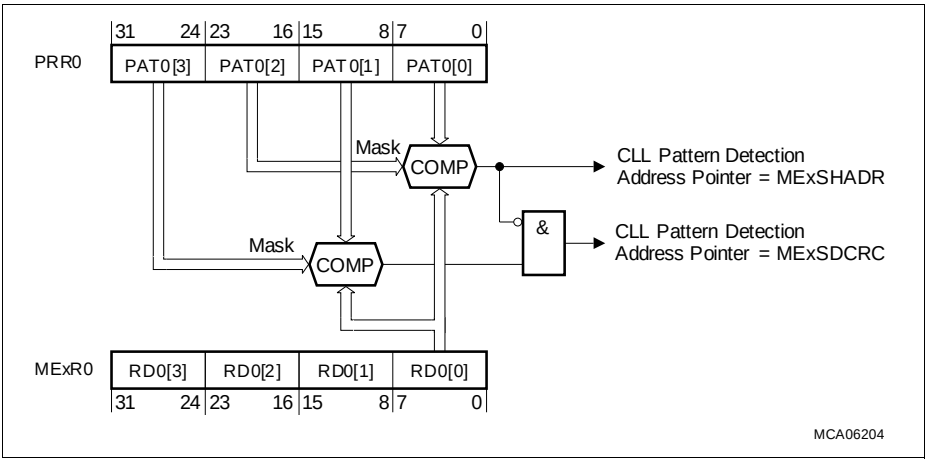


Figure 15-23 Conditional Linked List and Pattern Detection Logic

During each DMA move the pattern detection logic is used to determine the selection of the address pointer:

- If PAT[0] masked by PAT[2] is matching with the current read data MEx0R.RD0[0] then when the current DMA move is completed load a new transaction control set. The shadow address register MExSHADR stores the address pointer to the next transaction control set in the linked list.
- If PAT[1] masked by PAT[3] is matching with the current read data MEx0R.RD0[0] then when the current DMA move is completed load a new transaction control set. The source and destination address CRC checksum register MExSDCRC stores the address pointer to the next transaction control set in the linked list.
- If both PAT[0] masked by PAT[2] and PAT[1] masked by PAT[3] are matching with the current read data MEx0R.RD0[0] then when the current DMA move is completed

load a new transaction control set. The shadow address register MExSHADR stores the address pointer to the next transaction control set in the linked list.

- If there is no match then continue with the DMA transaction.

If a pattern match is detected then:

- The DMA transaction ends with the current DMA move.
- TSRz.CH will clear when the DMA write move has completed.

Completion of Current DMA Transaction

If the current DMA transaction completes when MExCHSR.TCOUNT = 0 and there is no pattern match then the conditional linked list will load a new transaction control set. The shadow address register MExSHADR stores the address pointer to the next transaction control set in the linked list.

Multiple Pattern Detection Conditions

The user may intentionally program PAT[0] masked by PAT[2] not to match with the current read data MEx0R.RD0[0] and transition across a series of DMA transactions in the Conditional Linked List. In each DMA transaction the user may programme a series of different PAT[1] masked by PAT[3] values and test if each value matches with the current read data MEx0R.RD0[0].

If PAT[1] masked by PAT[3] is matching with the current read data MEx0R.RD0[0] then when the current DMA move is completed load a new transaction control set. The source and destination address CRC checksum register MExSDCRC stores the address pointer to the next transaction control set in the linked list.

15.4.4 Transaction Control Engine

Each DMA sub-block has a Transaction Control Unit. The Transaction Control Unit in the DMA sub-block, as shown in the DMA Controller block diagram in **Figure 15-24**, contains the transaction control set stored in the active channel register set and a move engine.

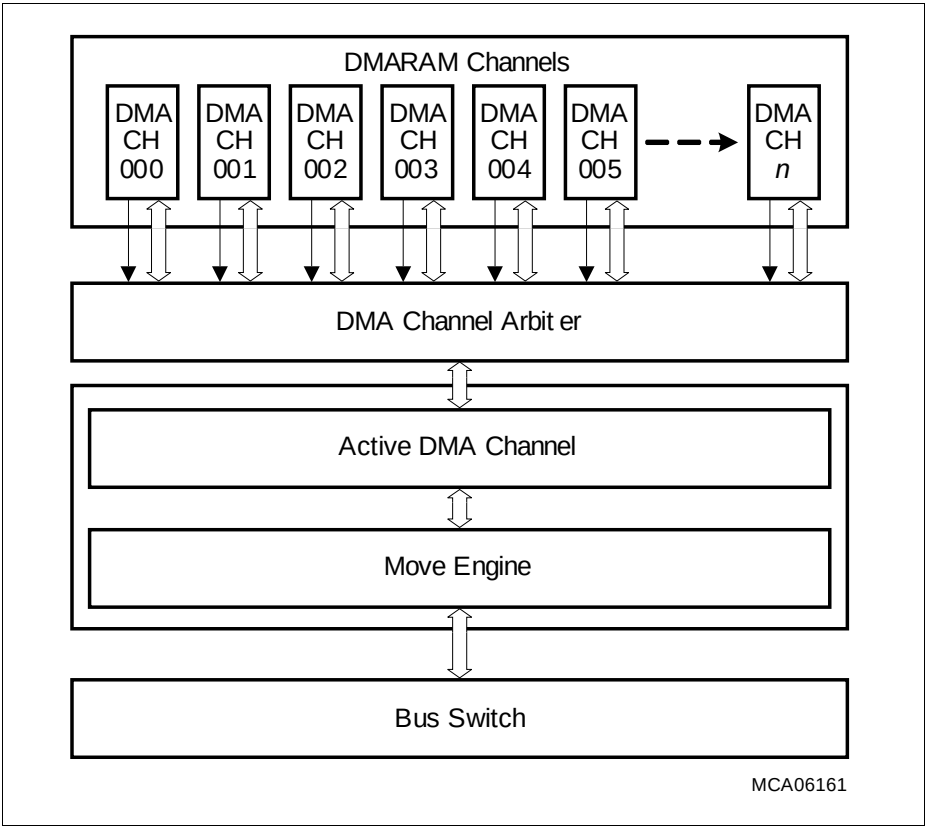


Figure 15-24 Transaction Control Engine

Any move engine can service hardware or software requests from any DMA channel.

DMA Channel Arbitration

The DMA channel arbiter continuously monitors DMA channel transfer access pending requests. The highest number DMA channel with an access pending wins the DMA channel arbitration. On winning arbitration the transaction control set is read from the

DMARAM and written to the DMA sub-block active channel register set. The DMA channel parameters are passed to the move engine and a DMA transfer composed of a number of DMA moves is processed.

On completion of each DMA transfer the move engine re-arbitrates for the highest priority DMA channel and services the next DMA transfer request. If there is a higher priority DMA transfer request then the current lower priority DMA channel transaction control set is copied to the DMARAM and the active channel register set overwritten with the parameters of the higher priority DMA channel.

If the DMA channel is halted, suspended or has finished the DMA transaction then the move engine will also re-arbitrate for the highest priority DMA request.

DMA Read Buffer

Each DMA move engine includes a 256-bit buffer to store read data from DMA read moves. The read buffer supports the reading of four double words (=256-bit) of data.

A read move to a cached area of memory (Segments 8 and 9) will be translated into an SRI-Bus BTR4 transaction. The 256-bit read data will be stored together with the related 64-bit aligned address in the move engine. If the next and subsequent read access to a segment 8 or 9 address is identical (64-bit aligned) to the current read buffer contents, the requested read data will be read from the read buffer instead of from the SRI-Bus.

If the next read from a segment 8 or 9 address is not identical (64-bit alignments) to the current read buffer contents, the content of the read buffer is invalidated. A BTR4 read request is generated, the read buffer will be updated with the new 256-bit data and its related addresses.

The loading of a new DMA transfer into the move engine or a DMA write to a segment 8 or 9 address invalidates the DMA read buffer.

Address range of segments 8 & 9 is $8000\ 0000_H$ - $9FFF\ FFFF_H$.

DMA Move Engine

The move engine requests the required buses and loads or stores data according to the parameters of a DMA transfer. It is able to wait if a targeted bus is not available.

The processing of a DMA transfer (composed of several DMA moves) of a DMA transaction by the move engine cannot be interrupted and is always completed. A DMA channel interrupt, reset, halt request or debug suspend only becomes active when the DMA channel completes the current DMA transfer.

On completion of a DMA transfer the move engine will send back the updated address register information to the active channel register set. If the current DMA channel continues to win arbitration then the move engine will continue processing DMA transfers until the DMA transaction is completed.

Possible source and destination error conditions are also reported.

15.4.4.1 Error Conditions

The DMA move engine reports bus error and source/destination error status. In the case of multiple errors, the error bits are set according to the error conditions. This means that more than one bus error flag and source/destination flag can be set.

The DMA channel should be configured to generate an error interrupt service request.

Bus Errors

The bus error flag ERRSRx.SPBER indicates an SPB Bus error occurred during a move (read or write) of a DMA module transaction.

The bus error flag ERRSRx.SRIER indicates an SRI Bus error that occurred during a move (read or write) of a DMA module transaction.

Source and Destination Errors

The source error flags ERRSRx.SER indicate that an error occurred during a DMA read move from a source address during a DMA transaction of DMA sub-block x.

The destination error flags ERRSRx.DER indicate that an error occurred during a DMA write move to a destination address during a DMA transaction of DMA sub-block x.

If a source or destination error is reported then the DMA transaction completes. If a source error is reported during a DMA read move then the DMA write move is not executed, but the destination address is updated.

Source and destination errors include unsupported types of bus transaction.

Source and Destination Errors in a Linked List

If a DMA channel is configured as a linked list and a source or destination error is reported then the current DMA transaction must complete. In the case of the Conditional Linked List pattern matches will be disabled. The new transaction control set must not be loaded and the linked list stops to allow debug of the current DMA transaction.

Transaction Control Set Load Error in a Linked List

During all linked list operations (DMA, Accumulated, Safe and Conditional) if an error is reported during the loading of a next transaction control set then the error flag ERRSRx.DLLER is set, the last error channel ERRSRx.LEC is recorded and the source of the error is set. The linked list operation will be aborted and the transaction request state bit TSRz.CH bit cleared.

Safe Linked List SDCRC Errors

A Safe Linked List must confirm that the calculated SDCRC checksum matches the expected SDCRC checksum before the next transaction control set is loaded. If the

SDCRC checksum does not match then a Safe Linked List error ERRSRx.SLLER is reported.

RAM Errors

When a DMA channel wins arbitration the transaction control set is copied from the DMARAM to the DMA sub-block active channel register set. If an ECC error is signalled during the DMARAM read then the error flag ERRSRx.RAMER is set and the last error channel ERRSRx.LEC recorded. The DMA transfer will be aborted and the transaction request state bit TSRz.CH bit cleared.

Transaction Request Lost

The transaction request lost error flag TSRz.TRL indicates if a DMA request for DMA channel z has been lost.

15.4.5 Bus Switch, Bus Switch Priorities

The bus switch of the DMA controller provides the connection from the DMA sub-blocks and system peripheral masters to on chip bus master interfaces (see **Figure 15-25**). One access can be buffered in the bus interfaces.

The Bus Switch supports Switch Address Routing to the whole memory range.

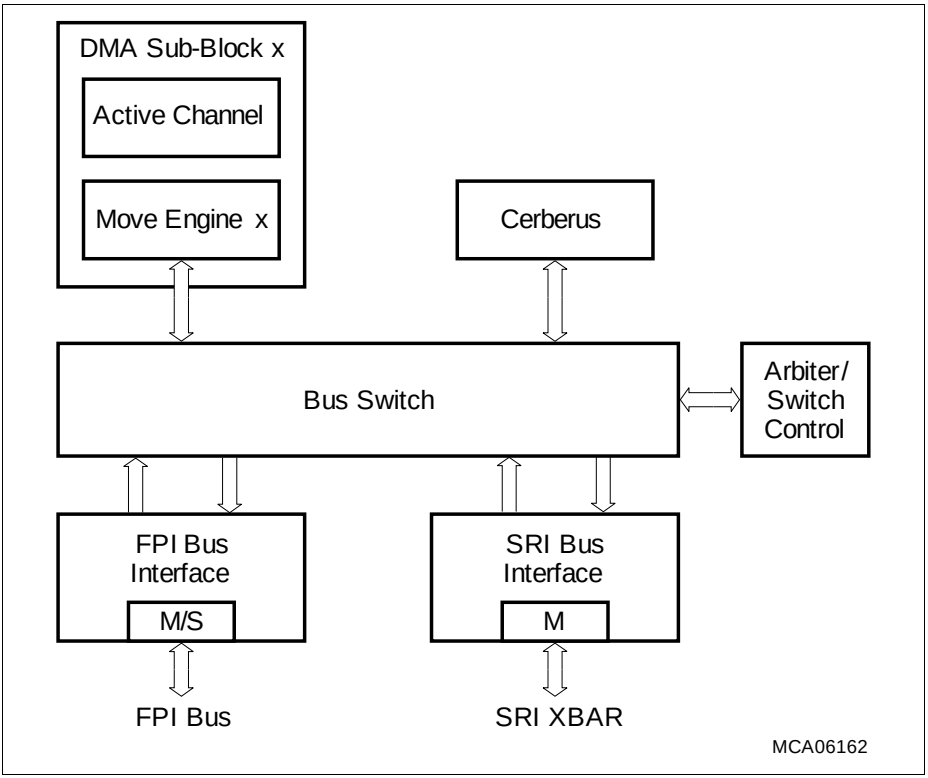


Figure 15-25 Bus Switch

Note: The accesses of the DMA Move Engine's bus interfaces to the On Chip Bus interfaces are always done in Supervisor Mode.

The arbiter/switch control unit arbitrates the requests from the connected active interfaces (DMA move engines and Cerberus) and grants the buses connected to the switch for data transfers. **Table 15-2** and **Table 15-3** defines the Bus Switch priorities for requests to the same on chip bus interface. The arbitration scheme is valid in case of a collision of requests from active peripherals connected to the DMA bus switch (move engines) for the same resource (SPB Bus Interface, SRI Bus Interface). The arbitration is done for each bus switch request.

In case of a collision between DMA move engine requests and a Cerberus request on the DMA bus switch for the same resource, the priorities are as given in **Table 15-2**.

Table 15-2 DMA Bus Switch Priorities

Priority	Agent Requests	Comment
Highest	Cerberus to On Chip Bus High	Priority selection by software in Cerberus.
	DMA move engine write	-The detailed Bus Switch priorities for the two engines with concurrent reads or concurrent writes are listed in **Table 15-3**.
	DMA move engine read	-The detailed Bus Switch priorities for the two engines with concurrent reads or concurrent writes are listed in **Table 15-3**.
Lowest	Cerberus to On Chip Bus Low	Priority selection by software in Cerberus.

Concurrent SRI Accesses

In case of a collision of both move engines with concurrent read requests or concurrent write requests for the SRI resource, the highest channel wins.

Concurrent SPB Accesses

In case of a collision of both move engines with concurrent read requests or concurrent write requests for the SPB resource, the move engine with the highest DMA priority determined by MExCHCR.DMAPRIO determines the winning arbitration on the DMA bus switch (see **Table 15-3**).

Table 15-3 DMA SPB Resource Switch Priorities of DMA Move Engines

Priority	DMA Move Engine Request	Comment
Highest	Move Engine x MExCHCR.DMAPRIO = “11”	Move engine with high DMA priority.
	Move Engine x MExCHCR.DMAPRIO = “10”	Move engine with medium DMA priority.
	Move Engine x MExCHCR.DMAPRIO = “01”	Move engine with medium DMA priority.
Lowest	Move Engine x MExCHCR.DMAPRIO = “00”	Move engine with low DMA priority.

If the move engines are processing DMA moves with concurrent read requests or concurrent write requests of equal DMA priority then the move engine processing the highest DMA channel number wins the arbitration at the DMA bus switch.

15.4.6 DMA Module Priorities on On Chip Busses

Every active peripheral connected to the DMA bus switch that requests for access to SPB Bus or SRI Bus has to go through two arbitration stages before accessing the on chip bus: DMA internal arbitration at the DMA bus switch and DMA module external arbitration at the on chip bus.

The DMA can be connected to the SPB Bus and to the SRI Bus with master interfaces.

- The complete list of SPB master priorities can be found in the SPB Bus Control Unit Chapter.
- The complete list of SRI master priorities can be found in the Shared Resource Interconnect Chapter.

15.4.6.1 On Chip Bus Access Rights, RMW support

All accesses triggered by the DMA Move Engines are always done in SV mode.

The DMA module does not support read/modify write instructions.

15.4.6.2 On Chip Bus Master Interfaces

This chapter describes the features of the DMA on chip bus master interfaces to the SPB Bus and to the SRI Bus.

The DMA SPB master interface supports:

- single data read and write transactions (8-bit, 16-bit, 32-bit)
- generation of interleaved FPI transactions from different sources (move engines)
- de-assertion of request after retry in order to prevent bus blocking.
- out of order transactions from different sources in order to avoid side effects (blocking) between the different sources (move engines)
- three dedicated SPB requests (low, medium, high priority)[1)].

A single move engine supports only one DMA transaction from one DMA channel at a time and generates a sequence of read - write sequences. The DMA sub-block will not generate permanent, pipelined, high priority requests.

The DMA SRI master interface supports:

- single data read and write transactions (8-bit, 16-bit, 32-bit, 64-bit)[2)]
- block transfer read and write transactions (128-bit, 256-bit)[3)]
- generation of interleaved SRI transactions from different sources (move engines)

1) The complete list of SPB master priorities can be found in the SPB Bus Control Unit Chapter.

2) 64-bit DMA move supported for DMA read move from SRI source to DMA write move to SRI destination

3) Block transfer DMA move supported for DMA read move from SRI source to DMA write move to SRI destination

A single move engine supports only one DMA transaction from one DMA channel at a time and generates a sequence of read - write sequences. The DMA sub-block will not generate permanent, pipelined, high priority requests.

15.4.7 Pattern Detection

Pattern detection is only supported for 8-bit, 16-bit and 32-bit channel data width. Only the read data in MEx0R is compared with the value in one of the pattern read registers.

After a DMA read move, read data in the move engine least significant word read register MEx0R can be compared with data stored in one of the pattern read registers. Any move engine can service any DMA channel request. If pattern detection is selected then the DMA channel must be configured to select one of the pattern read registers (PRR0 or PRR1) for the pattern compare.

The result of a pattern compare match is always stored in a bit (MExCHSR.LXO) of the channel status register of the active DMA channel in the move engine x that is currently executing the DMA move. Therefore, the pattern match result LXO of the previous read move can also be combined together with the pattern match result of the actual read move. Move engine x read register MEx0R is overwritten with each read move.

The configuration and capabilities of the pattern detection logic further depends on the settings of MExCHCR.CHDW. CHDW determines the data width for the DMA read and write moves for each individual DMA channel z. The control bits, MExCHCR.PATSEL, selects among the different pattern detection modes for a specific value of CHDW.

If a DMA channel z is configured for pattern detection then the appendage of timestamps is not supported (ADICRz.STAMP = 0_B).

Interrupt Service Request

If MExCHCR.PATSEL[1:0] is not equal to 00_B and a pattern match is detected then:

- The DMA transaction ends with the current DMA move.
- TSRz.HTRE is cleared to stop DMA transfers on the current active channel z.
- TSRz.CH will clear when the DMA write move has completed.
- A pattern detection interrupt service request will be raised corresponding to the current active DMA channel z except in the case that a source error is reported.

The value of MExCHSR.TCOUNT and MEx0R can be read out by the interrupt software.

If MExCHCR.PATSEL=000_B or 100_B then no action will be taken when a pattern match is detected. A wrap interrupt can be used.

The software will have to service the interrupt and to activate the channel again.

15.4.7.1 Pattern Compare Logic

Read move data and compare match patterns are compared on a bit-wise level. The logic as shown in **Figure 15-26** is implemented in each COMP block of **Figure 15-27**, **Figure 15-28**, and **Figure 15-29**. One COMP block controls either 8 bits or 16 bits of data and makes it possible to mask each data bit for the compare operation.

In the compare logic for one bit of the COMP block, a data bit from register MEx0R is compared to the corresponding pattern bit stored in a pattern read register. If both bits

are equal and a pattern mask bit stored in another part of pattern read register is 0, the compare matched condition becomes active. When the pattern mask bit is set to 1, the compare matched condition is always active (set) for the related bit. When the compare matched conditions for each bit within a COMP block are true, the compare match output line of the COMP block becomes active.

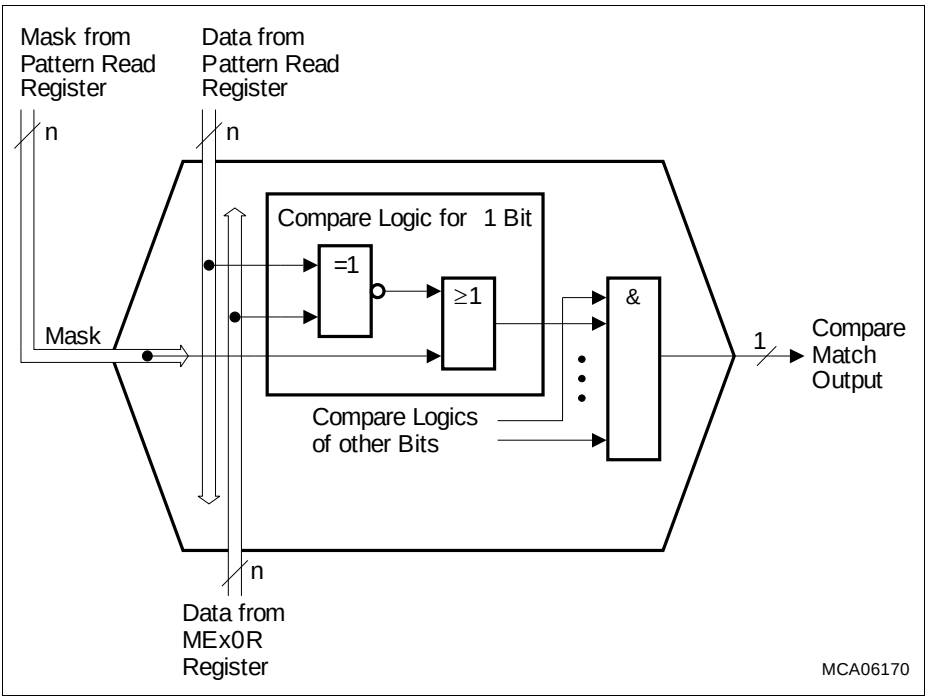


Figure 15-26 Pattern Compare Logic (COMP Block)

15.4.7.2 Pattern Detection for 8-bit Data Width

When 8-bit channel data width is selected (MExCHCR.CHDW = 000_B) and the channel is configured to select PRR0 (MExCHCR.PATSEL[2] = 0_B), the pattern detection logic is configured as shown in **Figure 15-27**.

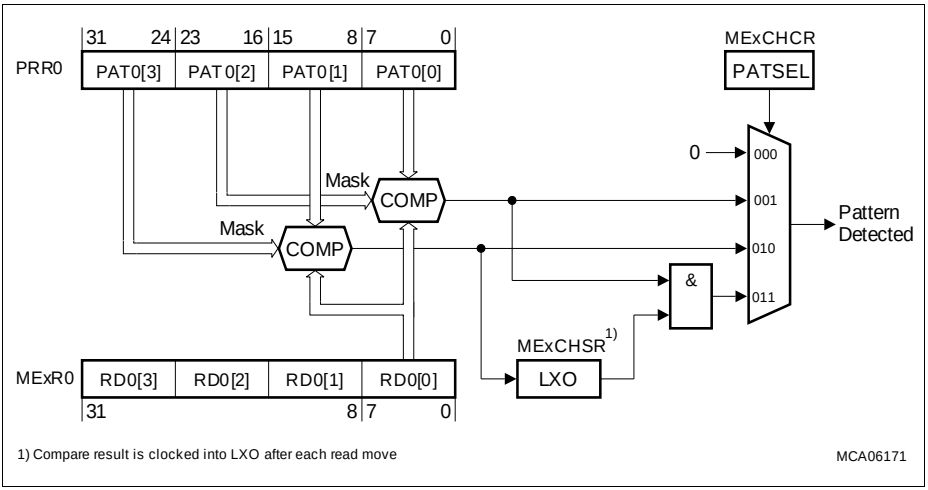


Figure 15-27 Pattern Detection for 8-bit Data Width (MExCHCR.CHDW = 000_B)

When 8-bit channel data width is selected, the pattern detection logic allows the byte of a DMA read move to be compared with two different patterns. Three compare match configurations are possible (see **Table 15-4**). A mask operation of each compared bit is possible.

Table 15-4 Pattern Detection for 8-bit Data Width

MExCHCR. PATSEL	Pattern Detection Operating Modes
000_B	Pattern detection disabled
001_B	Pattern compare of MExR.RD0[0] to PAT0[0], masked by PAT0[2]
010_B	Pattern compare of MExR.RD0[0] to PAT0[1], masked by PAT0[3]
011_B	Pattern compare of MExR.RD0[0] to PAT0[0], masked by PAT0[2] of the actual DMA read move **and** Pattern compare of MExR.RD0[0] to PAT0[1], masked by PAT0[3] of the previous DMA read move of DMA channel z
100_B	Pattern detection disabled
101_B	Pattern compare of MExR.RD0[0] to PAT1[0], masked by PAT1[2]

Table 15-4 Pattern Detection for 8-bit Data Width

MExCHCR. PATSEL	Pattern Detection Operating Modes
110_B	Pattern compare of MExR.RD0[0] to PAT1[1], masked by PAT1[3]
111_B	Pattern compare of MExR.RD0[0] to PAT1[0], masked by PAT1[2] of the actual DMA read move **and** Pattern compare of MExR.RD0[0] to PAT1[1], masked by PAT1[3] of the previous DMA read move of DMA channel z

Two Byte Pattern Detection Sequence

If the DMA channel pattern selection is configured to select a two byte sequence (MExCHCR.PATSEL[1:0] = 11_B) against PRR0 (MExCHCR.PATSEL[2] = 0_B) then after each DMA read move the pattern match result “MEx0R.RD0[0] with PRR0.PAT0[1], masked by PRR0.PAT0[3]” is stored in bit MExCHSR.LXO.

This operating mode allows, for example, two-byte sequences to be detected in an 8-bit data stream coming from a serial peripheral unit with 8-bit data width (e.g.: recognition of carriage-return, line-feed characters).

15.4.7.3 Pattern Detection for 16-bit Data Width

When 16-bit channel data width is selected (MExCHCR.CHDW = 001_B) and the DMA channel is configured to select to PRR0 (MExCHCR.PATSEL[2] = 0_B), the pattern detection logic can be configured as shown in **Figure 15-28**.

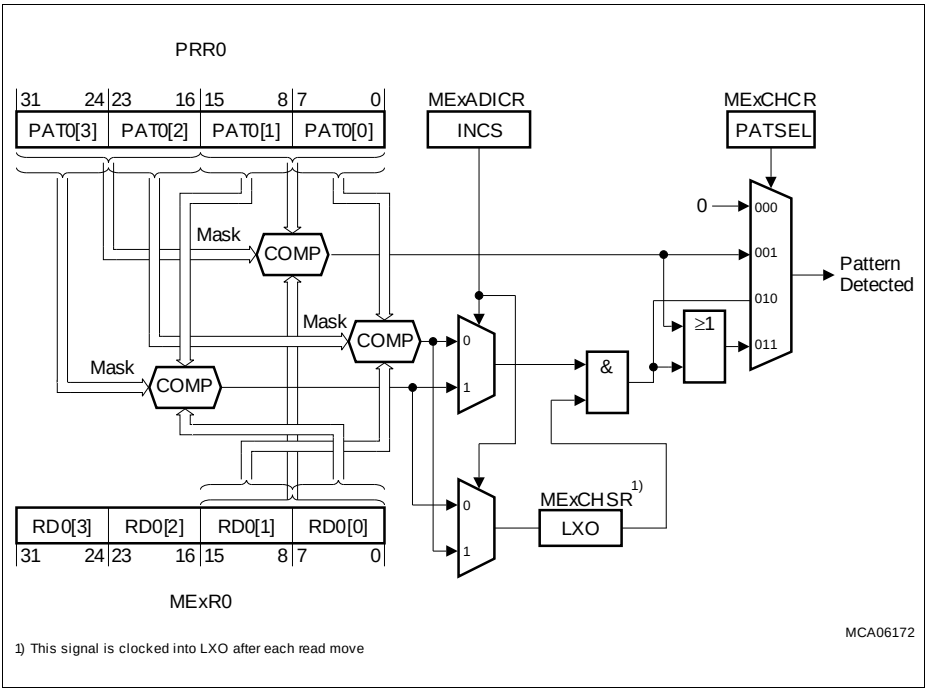


Figure 15-28 Pattern Detection for 16-bit Data Width (CHCFGRz.CHDW = 001_B)

When 16-bit channel data width is selected, the pattern detection logic makes it possible to compare the complete half-word of one read move only (aligned mode) or to compare upper and lower byte of two consecutive read moves (unaligned modes). Both modes can be combined (combined mode) too. Three compare match configurations are possible (see **Table 15-5**). A mask operation of each compared bit is possible.

Table 15-5 Pattern Detection for 16-bit Data Width

MExCHCR.PATSEL	MExADICR.INCS	Pattern Detection Operating Modes
000_B	–	Pattern detection disabled
001_B	–	**Aligned Mode:** Pattern compare of MEx0R.RD0[1:0] to PAT0[1:0], masked by PAT0[3:2]

Table 15-5 Pattern Detection for 16-bit Data Width (cont'd)

MExCHCR. PATSEL	MExADICR. INCS	Pattern Detection Operating Modes
010_B	0	**Unaligned Mode 1 (Source Address Decrement):** Pattern compare of MEx0RMEx1R.RD0[1] to PAT0[0], masked by PAT0[2] of the actual DMA read move **and** Pattern compare of MEx0R.RD0[0] to PAT0[1], masked by PAT0[3] (LXO) of the previous DMA read move of DMA channel z
	1	**Unaligned Mode 2 (Source Address Increment):** Pattern compare of MEx0R.RD0[0] to PAT0[1], masked by PAT0[3] of the actual DMA read move **and** Pattern compare of MEx0R.RD0[1] to PAT0[0], masked by PAT0[2] (LXO) of the previous DMA read move of DMA channel z
011_B	0 or 1	**Combined Mode:** Pattern compare for aligned mode (PATSEL = 001_B) **or** unaligned modes (PATSEL = 010_B)
100_B	–	Pattern detection disabled
101_B	–	**Aligned Mode:** Pattern compare of MEx0R.RD0[1:0] to PAT1[1:0], masked by PAT1[3:2]
110_B	0	**Unaligned Mode 1 (Source Address Decrement):** Pattern compare of MEx0R.RD0[1] to PAT1[0], masked by PAT1[2] of the actual DMA read move **and** Pattern compare of MEx0R.RD0[0] to PAT1[1], masked by PAT1[3] (LXO) of the previous DMA read move of DMA channel z
	1	**Unaligned Mode 2 (Source Address Increment):** Pattern compare of MEx0R.RD0[0] to PAT1[1], masked by PAT1[3] of the actual DMA read move **and** Pattern compare of MEx0R.RD0[1] to PAT1[0], masked by PAT1[2] (LXO) of the previous DMA read move of DMA channel z
111_B	0 or 1	**Combined Mode:** Pattern compare for aligned mode (PATSEL = 101_B) **or** unaligned modes (PATSEL = 110_B)

Unaligned Mode 1

In unaligned mode 1 (source address decremented), the high byte (MEx0R.RD01) of the current and the low byte (MEx0R.RD00) of the previous 16-bit DMA read move are compared.

Unaligned Mode 2

In unaligned mode 2 (source address incremented), the low byte (MEx0R.RD00) of the current and the high byte (MEx0R.RD01) of the previous 16-bit DMA read move are compared.

Combined Mode

If it is not known on which byte boundary (even or odd address) the 16-bit pattern to be detected is located, the combined mode must be used. This mode is the most flexible mode that combines the pattern search capability for aligned and unaligned 16-bit data searches.

15.4.7.4 Pattern Detection for 32-bit Data Width

When 32-bit channel data width is selected (MExCHCR.CHDW = 010_B) the pattern detection logic and the DMA channel is configured to select to PRR0 (MExCHCR.PATSEL[2] = 0_B), is configured as shown in **Figure 15-29**.

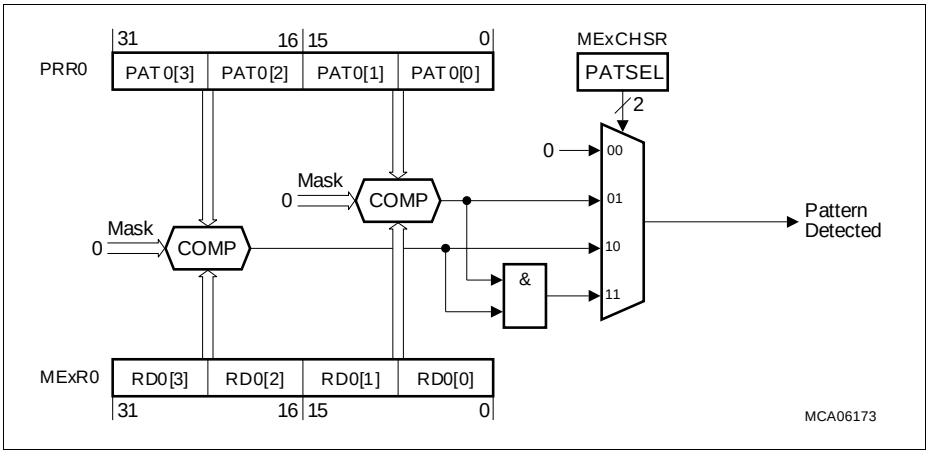


Figure 15-29 Pattern Detection for 32-bit Data Width (MExCHCR.CHDW = 010_B)

In 32-bit channel data width mode, the pattern detection logic makes it possible to compare the lower half-word only, the upper half-word only, or the complete 32-bit word

with a pattern stored in the pattern read register. Three compare match configurations are possible (see **Table 15-6**). A mask operation is not possible.

Table 15-6 Pattern Detection for 32-bit Data Width

MExCHCR.PATSEL	Pattern Detection Operating Modes
000_B	Pattern detection disabled
001_B	Unmasked pattern compare of RD0[1:0] to PAT0[1:0]
010_B	Unmasked pattern compare of RD0[3:2] to PAT0[3:2]
011_B	Unmasked pattern compare of RD0[3:0] to PAT0[3:0]
100_B	Pattern detection disabled
101_B	Unmasked pattern compare of RD0[1:0] to PAT1[1:0]
110_B	Unmasked pattern compare of RD0[3:2] to PAT1[3:2]
111_B	Unmasked pattern compare of RD0[3:0] to PAT1[3:0]

15.4.8 DMA Configuration Interface

The DMA controller implements a standard FPI slave interface compliant with the FPI bus protocol on the SPB bus. The SPB slave interface supports the following functions:

- Control of the DMA general control registers to support clock control, power management, debug, etc.
- Reading and writing DMA registers that manage DMA transactions.
- Reading and writing the transaction control set for each DMARAM channel z.
- Reading the active channel transaction registers in the DMA sub-block.
- Configuring and reading the move engine and error registers.

The DMA controller supports single data transfers and does not support block transfers.

15.4.8.1 DMARAM Channel Control and Status Word

The DMARAM Control and Status Word CHCSRz supports two functions:

- Storing channel status bits including the transfer count status. The status is updated when the active channel is written back on completion of a DMA transaction or on completion of a DMA transfer on losing channel arbitration.
- Write only triggers to set and clear state in the channel configuration and status.

The write only control and clear triggers (CHCSRz.SCH, TSRz.ECH, TSRz.DCH, CHCSRz.SWB, CHCSRz.CICH, CHCSRz.CWRP and CHCSRz.SIT) can be written independently of the channel transaction state (active, pending or idle).

15.4.8.2 DMA Active Channel Write Back

A DMA channel is active when a move engine services a hardware or software request. When a pending DMA channel request wins arbitration the transaction control set is copied from the DMARAM and loaded into the DMA sub-block active channel registers and the active channel field MExSR.CH is updated. The DMA channel is defined to be active.

The point at which transaction control set is written back to the DMARAM is dependent on the MExCHCR.RROAT:

- MExCHCR.RROAT = 0, the transaction control set is written back on completion of each DMA transfer.
- MExCHCR.RROAT = 1, the transaction control set is written back when the DMA channel loses channel arbitration and completes the current DMA transfer.

If a write back has occurred then the transaction control set stored in the DMARAM will not match the transaction control set held in the DMA sub-block active channel registers.

15.4.8.3 DMA Active Channel Shadow Control

An SPB write to a DMARAM channel z addresses other than the CHCSRz word when DMA channel z transaction request state bit TSRz.CH has an access pending will result in a bus error with the following shadow address source or destination address buffering exceptions as defined in **Table 15-7**.

Table 15-7 DMA Active Channel SPB Write Shadow Control Exceptions

SHCT	Description
0001_B	**Source Address Buffering (Read Only)** Active DMA channel z SPB writes to the source address SADRz are permitted. The shadow register MExSHADR in DMA sub-block x stores the SADRz value while DMA channel z is active in move engine x. On write back from DMA sub-block to DMARAM: • The source address SADRz stored in DMARAM is updated to the shadow address SHADRz value. • The shadow address SHADRz stored in DMARAM is set to 00000000_H.
0101_B	**Source Address Buffering (Direct Write)** Active DMA channel z SPB writes to the shadow address SHADRz are permitted and result in a update to the active channel shadow register MExSHADR in DMA sub-block x. On write back from DMA sub-block to DMARAM: • The shadow address SHADRz stored in DMARAM is updated on completion of the DMA transfer. • The source address SADRz is updated to the SHADRz value.
0010_B	**Destination Address Buffering (Read Only)** Active DMA channel z SPB writes to the destination address DADRz are permitted. The shadow register MExSHADR in DMA sub-block x stores the DADRz value while DMA channel z is active in move engine x. On write back from DMA sub-block to DMARAM: • The source address DADRz stored in DMARAM is updated to the shadow address SHADRz value. • The shadow address SHADRz stored in DMARAM is set to 00000000_H.
0110_B	**Destination Address Buffering (Direct Write)** Active DMA channel z SPB writes to the shadow address SHADRz are permitted and result in a update to the active channel shadow register MExSHADR in DMA sub-block x. On write back from DMA sub-block to DMARAM: • The shadow address SHADRz stored in DMARAM is updated on completion of the DMA transfer. • The destination address DADRz is updated to the SHADRz value.

15.4.8.4 DMARAM Write Back During Linked List Execution

When a linked list is executed by move engine x the contents of the move engine read register are selectively written back to the DMARAM transaction control set as defined in **Table 15-8**.

Table 15-8 Linked List Write Backs

SHCT	Description
1100_B	**DMA Linked List** The DMA controller reads a DMA channel transaction control set and overwrites 8 X words in the DMARAM. *Note: The SDCRC and RDCRC checksums are unique for each DMA transaction in the DMA LInked List.*
1101_B	**Accumulated Linked List** The DMA controller reads a DMA channel transaction control set and overwrites 6 X words in the DMA RAM. The SDCRC and RDCRC words are not overwritten. *Note: The SDCRC and RDCRC checksums are accumulated across all DMA transactions of the DMA Linked List.*
1110_B	**Safe Linked List** The DMA controller reads a DMA channel transaction control set. The Linked List only proceeds with the next DMA transaction if the existing SDCRC checksum matches the expected SDCRC checksum in the loaded from the new DMA transaction control set. *Note: The SDCRC and RDCRC checksums are accumulated across all DMA transactions of the DMA LInked List.*
1111_B	**Conditional Linked List** Shadow address register (MExSHADR) and source and destination address CRC register (MExSDCRC) are used as address pointers to a DMA linked lists. The selection of the address pointer is determined by DMA channel pattern detection conditions. *Note: Calculation of SDCRC checksums is not supported.*

15.4.9 Interrupt Service Requests

The interrupt structure of the DMA controller is very flexible. There are five different types of interrupt events. The service control registers are located in the Interrupt Router.

- Each DMA channel has its own interrupt service request that covers DMA channel traffic management events:
  - Channel transfer interrupt service request
  - Channel pattern detection interrupt service request
  - Channel source and destination wrap buffer interrupt service requests
- The DMA controller has one interrupt service request covering all error events:
  - Channel transaction request lost interrupt service request
  - Channel safe linked list SDCRC checksum comparison error
  - Move engine source and destination error interrupt service request
  - Read DMARAM ECC error interrupt service request
  - DMA linked list transaction control set load error interrupt service request

The DMA controller interrupt service requests are shown in **Figure 15-30**. The number of interrupt service requests in the DMA controller is one greater than the number of DMA channels.

The channel interrupt service requests detect normal DMA traffic events and raise an interrupt request via the shared DMA channel interrupt service request output. The interrupt event is internally generated as a request pulse and always stored in the active DMA Channel Status Register MExCHSR.

The move engine interrupt service requests detect error conditions and raise an interrupt request via the shared error interrupt service request output. Source and destination errors are stored in the move engine error status register ERRSRx. The transaction request lost condition is stored in the channel status register TSRz.TRL. A transaction request lost error interrupt service request is generated if ADICRz.ETRL enables the service request when channel z becomes an active DMA channel.

The routing of both channel transaction request lost and move engine source and destination error conditions via the error interrupt service request centralizes the reporting of all errors in each DMA sub-block to one interrupt service request. In the event that a DMA error interrupt request is raised then the error handler will read the contents of the move engine error status register ERRSRx and all the TSRz.TRL bits to identify the source of the error.

Software Activation of Interrupt Service Requests

Each channel interrupt service request can be activated by programming CHCSRz.SIT.

The error interrupt service request can be activated by programming ERRINTR.SIT.

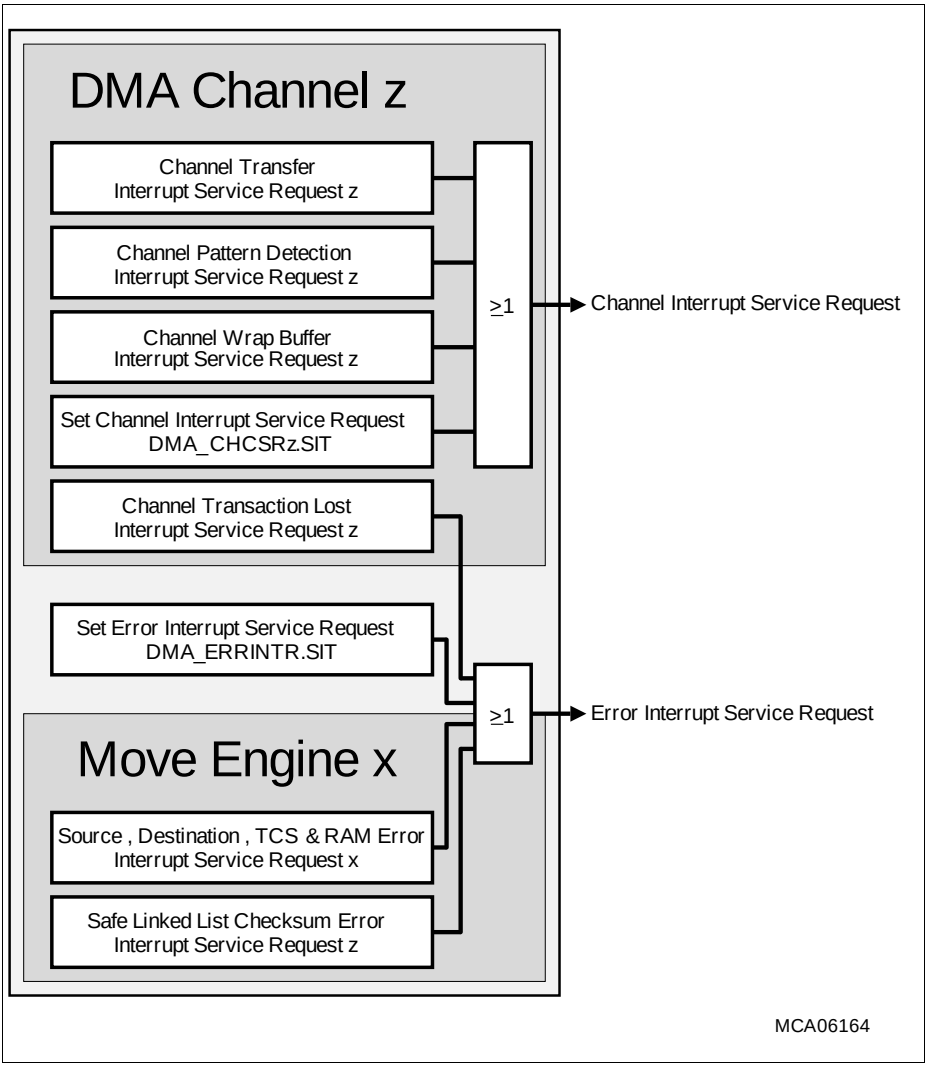


Figure 15-30 DMA Controller Interrupt Service Requests

The following sections describe each of the interrupt service request types in detail.

15.4.9.1 Channel Transfer Interrupt Service Request

The channel transfer interrupt service request is always activated after a DMA transfer, or when MExCHSR.TCOUNT matches with the value of bit field MExADICR.IRDV after it has been decremented after a DMA transfer. A channel transfer interrupt is indicated when status flag MExCHSR.ICH (CHCSRz.ICH) is set. The status flag can be reset together by software when setting bit CHCSRz.CICH (or TSRz.RST). The channel interrupt of DMA channel z is enabled when bit ADICRz.INTCT[1] is set.

Bit MExADICR.INTCT[0] selects one of two types of interrupt source. For the compare operation, bit field MExADICR.IRDV (4-bit) is zero-extended to 14-bit and then compared with the 14-bit TCOUNT value. This means that a TCOUNT match interrupt can be generated after one of the last 16 DMA transfers of a DMA transaction. Note that with IRDV = 0000_B, the match interrupt is generated at the end of a DMA transaction (after the last DMA transfer).

If MExCHCR.PRSEL = 1B for DMA channel z then the channel transfer service request interrupt is masked and the access pending bit is set in the next lower priority DMA channel z-1.

15.4.9.2 Channel Pattern Detection Interrupt Service Request

The channel pattern detection interrupt service request is enabled when MExCHCR.PATSEL[1:0] is not equal 00_B. The pattern detection interrupt is indicated when status flag MExCHSR.IPM is set. The status flag MExCHSR.IPM can be reset together by software when setting bit CHCSRz.CICH (or TSRz.RST).

Further details on the pattern detection are described in **Section 15.4.7**.

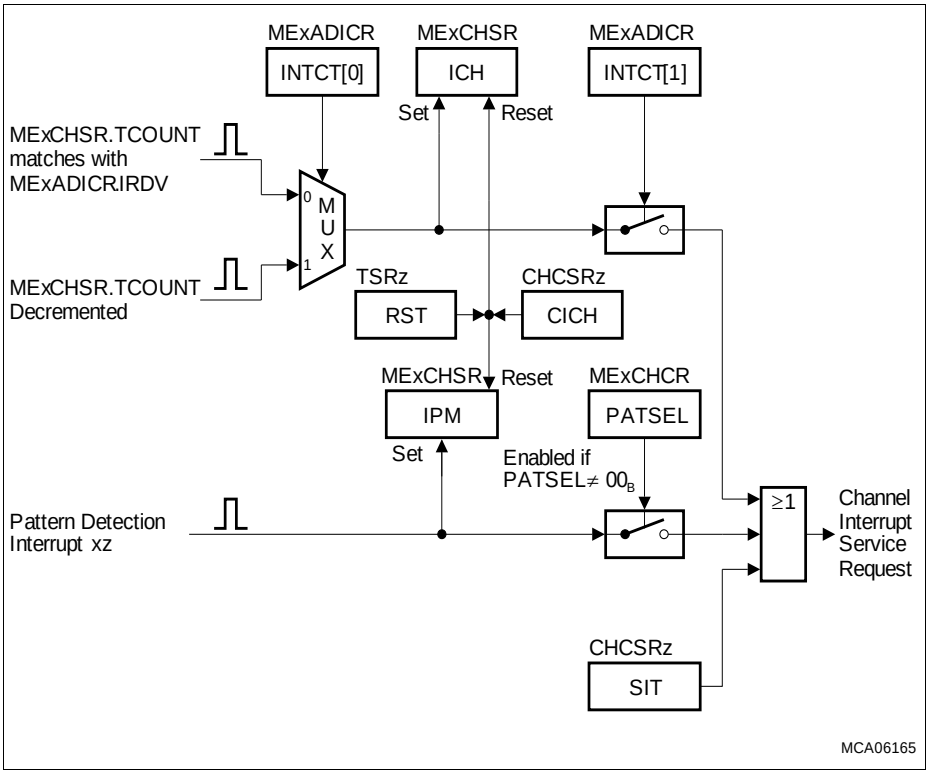


Figure 15-31 Channel Transfer and Pattern Detection Interrupt Service Requests

Figure 15-31 shows the implementation of the channel transfer interrupt service request, the pattern detection interrupt service request and the channel set interrupt service request functions.

15.4.9.3 Channel Wrap Buffer Interrupt Service Request

Each DMA channel z is able to generate a wrap buffer interrupt for source buffer or destination buffer overflow.

A wrap source buffer interrupt of DMA channel z is indicated by status flag MExCHSR.WRPS. A wrap destination buffer interrupt of DMA channel z is indicated by the status flag MExCHSR.WRPD. Both interrupt status flags can be reset by software when bit CHCSRz.CWRP (or TSRz.RST) becomes set. The wrap source buffer interrupt is enabled when bit MExADICR.WRPSE is set. The wrap destination buffer interrupt is enabled when bit MExADICR.WRPDE is set. The two interrupts for wrap source buffer and wrap destination buffer are OR-ed together with other channel interrupt service requests to form one common channel interrupt service request.

The channel pattern match detection must not be enabled while a wrap interrupt is enabled for the same channel.

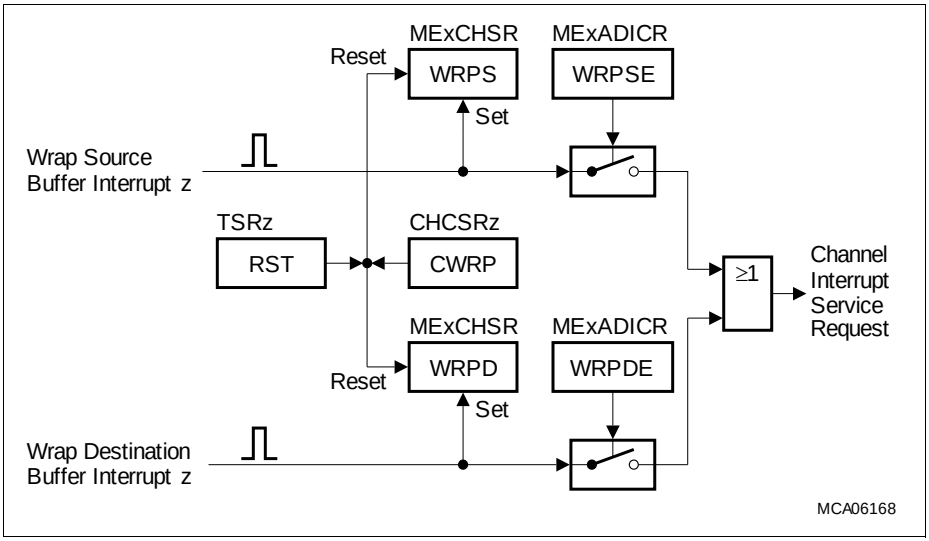


Figure 15-32 Channel Wrap Buffer Interrupt Service Requests

15.4.9.4 Transaction Request Lost Interrupt Service Request

Each DMA channel z is able to detect a transaction request lost condition. This condition becomes true when a new hardware or software DMA request occurs while the previous transaction or transfer on DMA channel z is not finished, indicated by TSRz.CH is still set. If such a transaction request lost condition occurs then bit TSRz.TRL is set.

A transaction request lost condition of DMA channel z is indicated by status flag TSRz.TRL, which can be reset by setting bit TSRz.CTL or TSRz.RST.

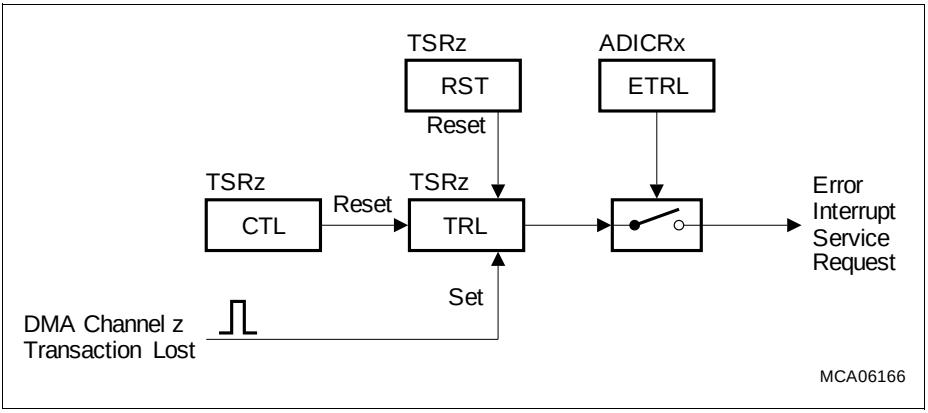


Figure 15-33 Transaction Request Lost Interrupt Service Requests

Interrupt Service Request

The transaction request lost interrupt service request is for DMA channel z is enabled when bit ADICRz.ETRL is set. The interrupt service request is routed via the DMA error service request.

15.4.9.5 Source and Destination Error Interrupt Service Requests

The Move Engine is able to detect error conditions that occur during accesses to the bus interfaces of the Bus Switch (see **Figure 15-25**). Two error conditions can be detected:

- Source error indicates a bus error occurred during a read move from the data source.
- Destination error indicates a bus error that occurred during a write move to the data destination.

A source error of Move Engine x is indicated by the status flag ERRSRx.SER. Status flag ERRSRx.SER can be reset by software when setting bit CLREx.CSER. The source error interrupt of Move Engine x is enabled when bit EERx.ESER is set. Separate reset, status, and enable bits are available in the Move Engines for source error condition, as well as for destination error condition.

Note that in case of a read move error, the write move is not executed but the destination address is updated.

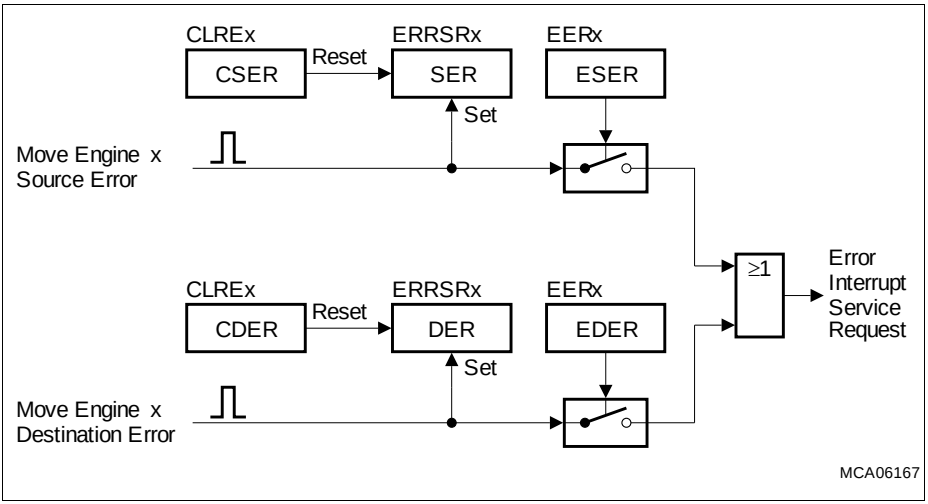


Figure 15-34 Move Engine Interrupt Service Requests

When a move engine source or destination error occurs, additional status bits and bit fields are provided in the error status register ERRSRx to indicate the following two status conditions:

- At which on chip bus interface a move engine error occurred.
- For which DMA channel z a move engine read or write move error was reported (ERRSRx.LEC).

These error status bits and bit fields are required by error handler software to detect in detail at which on chip bus interface and DMA channel z the move engine error has been generated.

- ERRSRx.SPBER is reset when bit CLREx.CSPBER is set.
- ERRSRx.SRIER is reset when bit CLREx.CSRIER is set.

The move engine interrupt service request can be activated by programming ERRINTR.SIT.

15.4.9.6 DMA Linked List Error Interrupt Service Request

During all DMA linked list operations the move engine is able to detect an error if the next transaction control set is not loaded correctly. This condition becomes true when a bus error is reported.

A DMA linked list error is indicated by the error status flag ERRSRx.DLLER. Status flag ERRSRx.DLLER can be reset by software when setting bit CLREx.CDLLER. Additional status bits are provided to indicate the following:

- At which on chip bus interface a move engine error occurred.
- For which DMA channel z the error was reported (ERRSRx.LEC).

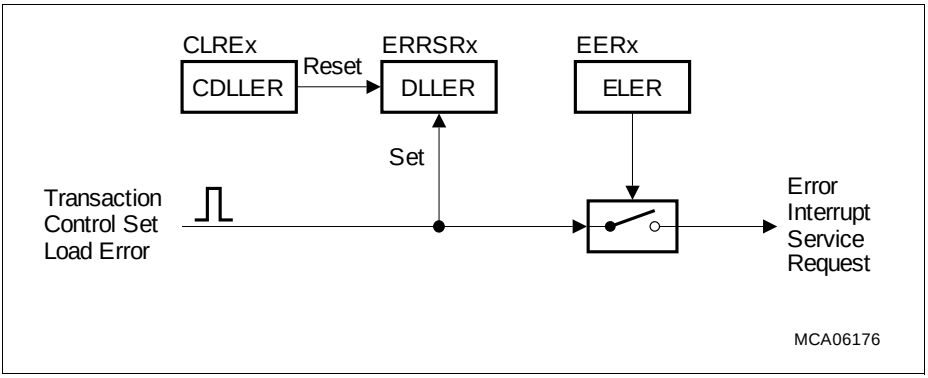


Figure 15-35 DMA Linked List Error Interrupt Service Requests

Interrupt Service Request

The DMA linked list error interrupt service request is enabled when bit EERx.ELER is set. The interrupt service request is routed via the DMA error service request.

15.5 Power Modes

15.5.1 Sleep Mode

Sleep Mode is enabled by the DMA Clock Control Register active low Sleep Mode Enable Control bit DMA_CLC.EDIS. If the enable bit is active when the sleep request input is activated then the DMA clock will be switched off when the move engines have completed any pending DMA transfers.

15.6 Functional Safety Features

15.6.1 Access Protection

A slave destination controls access to its bus peripheral interfaces and kernel address registers. Each on chip resource with bus master capability has a unique master tag identifier that is used to identify the source of an on chip bus transaction. The master tag identifier based access protection is used to enable write accesses to individual slave address ranges. There is no master tag identifier access control associated with read accesses.

Each DMA channel z including its transaction control set stored in DMARAM is assigned to one of y hardware resource partitions (default 0) and is write access control protected. Write accesses to DMA channel z control registers are only possible if the on chip bus master performing the write access has its master tag identifier enabled by the assigned hardware resource partition. Groups of DMA channels can be assigned to a hardware resource and write accesses are limited to individual on chip bus masters.

DMA Master Tag Identifiers

Each hardware resource partition supports a unique master tag identifier as shown in **Table 15-9**. The master tag identifier is driven onto the bus during a read or write access and allows the bus controller to identify the hardware resource requesting the access.

Table 15-9 Bus Master Tag Identifiers

Hardware Partition	Master Tag Identifier
DMA Hardware Resource 0	000110
DMA Hardware Resource 1	000111
DMA Hardware Resource 2	001000
DMA Hardware Resource 3	001001

DMA Supervisor/User Mode

The supervisor or user mode setting for a hardware resource is determined by the control bit MODEy.MODE. All the DMA channels assigned to a hardware resource assume the hardware resource supervisor or user mode setting. If the DMA bus master accesses a supervisor mode protected slave interface in user mode then the slave will reply with ERR acknowledge. The response will be recorded in the Move Engine x Error Status Register.

DMA Write Move

A DMA channel is programmed to read data from a source address and write the data to a destination address. For a DMA write move operation to successfully complete the slave access control logic must enable write accesses to the destination address for the bus master tag identifier of the requesting DMA hardware resource.

15.6.2 Data Integrity

The following data integrity error conditions are detected in the DMA and reported to the SMU:

- Error Correcting Code (ECC) errors generated by the DMARAM.
- Error Correcting Code (ECC) errors generated from SRI data.

15.6.2.1 DMARAM

The DMA controller detects several different classes of ECC error. These errors will set a status flag in the MEMCON register and send a trigger to the SMU for the processing of the error condition. ECC errors are reported for the following error conditions:

Internal ECC Error

The DMARAM is accessed by the internal DMA controller logic during the loading of the transaction control set into the DMA sub-block. If an ECC error occurs then the MEMCON.INTERR is set and a trigger is sent to the SMU. The DMA transaction will not take place and the channel number will be recorded in the move engine last error channel bit field ERRSRx.LEC and the error status bit ERRSRx.RAMER set.

SPB Read Access

If an ECC error is signalled during an SPB read access to the DMARAM then the MEMCON.DATAERR status bit will be set and a trigger sent to the SMU.

SPB Write Access

An SPB write access will require the DMARAM to perform an internal Read Modify Write (iRMW). If an ECC error is reported by the DMARAM during the read phase then the

MEMCON.RMWERR bit will be set and a trigger sent to the SMU. The write phase of the iRMW will not take place.

15.6.2.2 DMA SRI Read and Write Data

The SRI Bus protocol supports the reliable delivery of data between sources and destinations by extending the protocol to include ECC. The ECC word is generated across the address and data words. The DMA read move supports data integrity checking for data sourced from SRI Bus destinations. The ECC word is recoded for a subsequent DMA Write Move to an SRI Bus destination address.

ECC errors can be generated at two locations in the DMA:

- The DMA checks the integrity of read data received across the SRI Bus from SRI sources during a DMA read move.
- Write data to SRI destinations will be sourced from the Move Engine Read Register and is checked for data integrity during a DMA write move.

System Peripheral Bus

The SPB-Bus protocol does not support data integrity checking. ECC extensions will be generated when the read data is stored in the Move Engine Read Register. The integrity of SPB sourced data will be checked when it is written to the SRI Bus.

Data Integrity Testing

The alarm can also be triggered by software. The MBIST is used to generate an ECC error condition in a memory used as a DMA source. When the data is passed through the DMA it will generate an ECC error condition. See MTU chapter for details.

15.7 Debug Features

The DMA controller supports the following debugging capabilities:

- Channel suspend
- OCDS Trigger Bus
- MCDS Trace Interface

15.7.1 Channel Suspend Mode

The on chip debug control unit is able to generate a channel suspend mode request (SUSREQ) for the DMA controller. When this channel suspend request becomes active, the state of a DMA channel becomes frozen regarding hardware changes to ensure that the state of the DMA channels can be analyzed by reading the register contents. Pending transfers in the DMA module on chip Bus master interfaces (SPB master interface, SRI master interface) are completed. The DMA controller signals its channel suspend mode back to the on chip debug control via a suspend acknowledge.

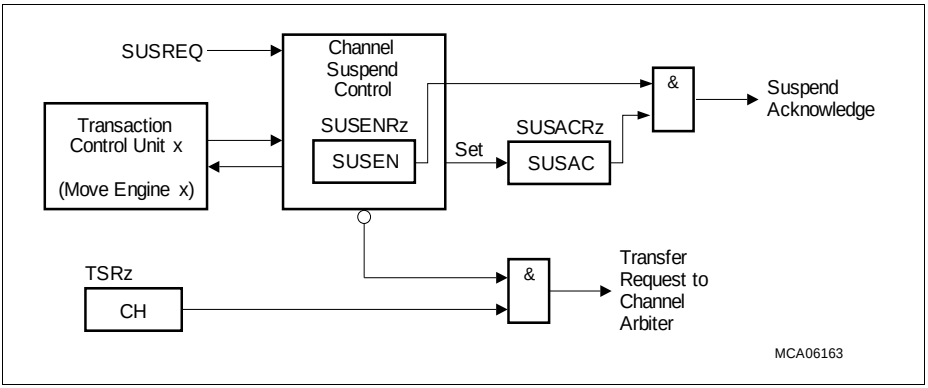


Figure 15-36 Channel Suspend Mode Control

Entering Channel Suspend Mode

Channel suspend mode of DMA channel z is entered if its suspend enable bit in the Suspend Mode Enable Register SUSENRz.SUSEN is set. When SUSREQ becomes active, the operation of all DMA channels enabled for channel suspend mode is stopped automatically after their current DMA transfer has finished in the transaction control unit. Afterwards, the suspend active status flag SUSACRz.SUSAC is set, indicating that DMA channel z is in channel suspend mode.

The DMA suspend acknowledge becomes active when all DMA channels z that are enabled for the Channel Suspend Mode have set their suspend active status flag SUSACRz.SUSAC. In Channel Suspend Mode, register contents can be modified. These modifications are taken into account for further DMA transactions or DMA transfers of the related DMA channel after Suspend Mode has been left again.

DMA channels that are disabled for Suspend Mode (SUSENRz.SUSEN = 0) continue with its normal operation.

Exiting Channel Suspend Mode

Channel suspend mode of DMA channel z is left and its normal operation continues if either the SUSREQ signal becomes inactive, or if the enable bit SUSENRz.SUSEN is reset by software.

15.7.2 OCDS Trigger Bus (OTGB) Interface

A Trigger Set is a collection of signals which supports a specific debug use case. The OCDS Trigger Set Select Register controls which Trigger Set is applied to the bus. The register is cleared by Debug Reset and by each System Reset when OCDS is disabled.

It is not touched by the System Reset when OCDS is enabled. The trigger sets are routed by the OCDS Trigger Mux (OTGM) to the OCDS Trigger Switch (OTGS) and the MCDS.

A DMA Trigger Set is unique to each DMA module dependent on the number of channels and move engines.

15.7.3 MCDS Trace Interface

The DMA module provides two 8 bit vectors for debug and trace signal generation purposes that are used for OCDS Level 3 and for OCDS Level 1:

- One 8 bit vector from SPB master interface via BCU_SPB to BAL_SPB inside MCDS (i.e. spb_clk domain).
- One 8 bit vector from SRI master interface via SRI_XBAR to BAL_SRI inside MCDS (i.e. sri_clk domain).

For each DMA master interface trace is captured to identify the DMA move engine and channel making the request.

Table 15-10 Trace Vector Definition

Bit	Trace
[6:0]	DMA Channel
[7]	DMA Move Engine 0_B Move Engine 0 1_B Move Engine 1

DMA Trace Signal Generation for OCDS Level 3

The trace mechanism allows the identity of the DMA move engine and channel accessing the on chip bus to be determined. The mechanism enables the MCDS system to trace transactions on the on chip bus generated by one or a group of dedicated DMA internal sources.

DMA Trace Signal Generation for OCDS Level 1

For OCDS Level 1 purposes the DMA providestwo 8 bit vectors.

15.8 Register Description

Figure 15-37 and **Table 15-12** show all registers associated with the DMA Controller Kernel. All DMA kernel register names described in this section are also referenced in other parts of the TC21x/TC22x/TC23x User's Manual by the module name prefix "DMA_".

DMA Register Index

The following index numbers are used:

- Index "x" is used to refer to the DMA sub-block (x = 0-1).
- Index "y" is used to refer to the hardware resource partitions (y = 0-3).
- Index "z" is used to refer to the DMA channel (z = 000-015).

DMA Register Classes

The registers fall into the following distinctive classes:

- Control registers that support clock control and the access protection to all registers and the DMARAM.
- Channel control registers are DMA channel control bits directly written into registers:
  - Channel reset
  - Hardware transfer request enable
  - Access pending set by either hardware triggers or software transfer requests
  - Transaction request lost status
  - Suspend enable
  - Suspend acknowledge
  - 2-bit hardware resource to partition channels across DMA partitions
- Active channel control read only registers store DMA channel control and status information for the current DMA move performed by the sub-block move engine.
- Move engine and error registers store control, status and read data for the current DMA transaction.

DMA Registers Overview

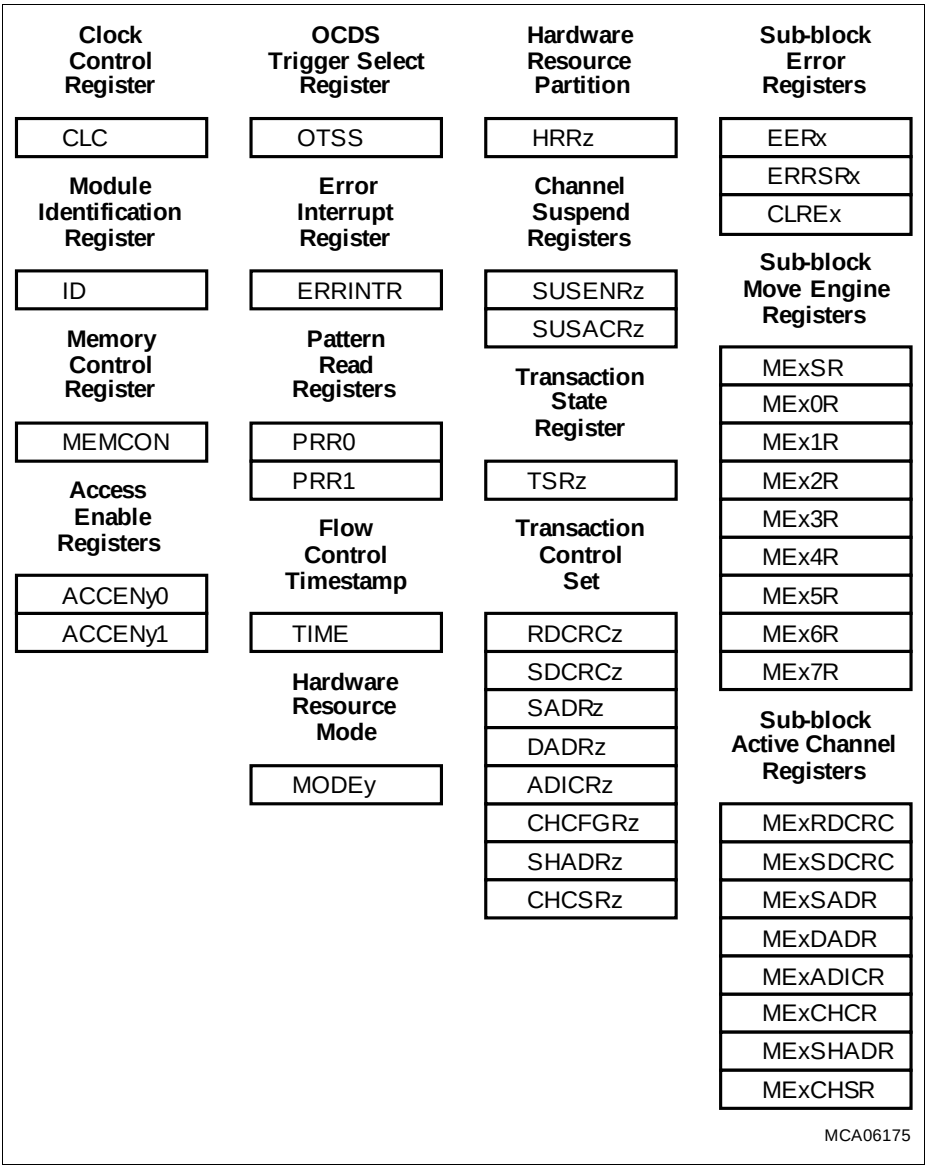


Figure 15-37 DMA Kernel Registers (x = 0-1, y = 0-3, z = 000-015)

Table 15-11 Registers Address Space - DMA Registers

Module	Base Address	End Address	Note
DMA	$F001\ 0000_H$	$F001\ 3FFF_H$	DMA Registers

Table 15-12 Registers Overview - DMA Registers

Short Name	Description	Offset Addr.[1]	Access Mode		Reset Class	Description See
			Read	Write		
DMA_CLC	DMA Clock Control Register	0000_H	U, SV	SV, E, P00, P01[2]	3	Page 15-86
-	Reserved	0004_H	BE	BE	-	-
DMA_ID	DMA Module Identification Register	0008_H	U, SV	BE	-	Page 15-87
-	Reserved	$000C_H$ - $001C_H$	BE	BE	-	-
DMA_MEMCON	DMA Memory Control	0020_H	SV, 32	SV, E, P00, P01, 32	3	Page 15-88
-	Reserved	0024_H - $003C_H$	BE	BE	-	-
DMA_ACCENy0	DMA Hardware Resource y Access Enable Register 0 (y = 0-3)	0040_H + (y * 8_H)	U, SV	SV, SE	3	Page 15-90
DMA_ACCENy1	DMA Hardware Resource y Access Enable Register 1 (y = 0-3)	0044_H + (y * 8_H)	U, SV	SV, SE	3	Page 15-91
-	Reserved	0060_H - $011C_H$	BE	BE	-	-
DMA_EERx	DMA Enable Error Register x (x = 0-1)	0120_H + (x * 1000_H)	U, SV	SV	3	Page 15-92
DMA_ERRSRx	DMA Error Status Register x (x = 0-1)	0124_H + (x * 1000_H)	U, SV	BE	3	Page 15-94
DMA_CLREx	DMA Clear Error Register x (x = 0-1)	0128_H + (x * 1000_H)	U, SV	SV	3	Page 15-97

Table 15-12 Registers Overview - DMA Registers

Short Name	Description	Offset Addr.[1)]	Access Mode		Reset Class	Description See
			Read	Write		
-	Reserved (x = 0-1)	$012C_H$ + (x * 1000_H)	BE	BE	-	-
DMA_MExSR	DMA Move Engine x Status Register (x = 0-1)	0130_H + (x * 1000_H)	U, SV	BE	3	Page 15-99
-	Reserved (x = 0-1)	0134_H + (x * 1000_H) - $013C_H$ + (x * 1000_H)	BE	BE	-	-
DMA_MEx0R	DMA Move Engine x Read Register 0 (x = 0-1)	0140_H + (x * 1000_H)	U, SV	BE	3	Page 15-100
DMA_MEx1R	DMA Move Engine x Read Register 1 (x = 0-1)	0144_H + (x * 1000_H)	U, SV	BE	3	Page 15-101
DMA_MEx2R	DMA Move Engine x Read Register 2 (x = 0-1)	0148_H + (x * 1000_H)	U, SV	BE	3	Page 15-102
DMA_MEx3R	DMA Move Engine x Read Register 3 (x = 0-1)	$014C_H$ + (x * 1000_H)	U, SV	BE	3	Page 15-103
DMA_MEx4R	DMA Move Engine x Read Register 4 (x = 0-1)	0150_H + (x * 1000_H)	U, SV	BE	3	Page 15-104
DMA_MEx5R	DMA Move Engine x Read Register 5 (x = 0-1)	0154_H + (x * 1000_H)	U, SV	BE	3	Page 15-105
DMA_MEx6R	DMA Move Engine x Read Register 6 (x = 0-1)	0158_H + (x * 1000_H)	U, SV	BE	3	Page 15-106

Table 15-12 Registers Overview - DMA Registers

Short Name	Description	Offset Addr.[1)]	Access Mode		Reset Class	Description See
			Read	Write		
DMA_MEx7R	DMA Move Engine x Read Register 7 (x = 0-1)	$015C_H$ + (x * 1000_H)	U, SV	BE	3	Page 15-107
-	Reserved (x = 0-1)	0160_H + (x * 1000_H) - $017C_H$ + (x * 1000_H)	BE	BE	-	-
DMA_MExRDCRC	DMA Move Engine x Channel Read Data CRC Checksum Register (x = 0-1)	0180_H + (x * 1000_H)	U, SV	BE	3	Page 15-129
DMA_MExSDCRC	DMA Move Engine x Channel Source and Destination Address CRC Checksum Register (x = 0-1)	0184_H + (x * 1000_H)	U, SV	BE	3	Page 15-128
DMA_MExSADR	DMA Move Engine x Source Address Register (x = 0-1)	0188_H + (x * 1000_H)	U, SV	BE	3	Page 15-127
DMA_MExDADR	DMA Move Engine x Channel Destination Address Register (x = 0-1)	$018C_H$ + (x * 1000_H)	U, SV	BE	3	Page 15-126
DMA_MExADICR	DMA Move Engine x Channel Address and Interrupt Control Register (x = 0-1)	0190_H + (x * 1000_H)	U, SV	BE	3	Page 15-116

Table 15-12 Registers Overview - DMA Registers

Short Name	Description	Offset Addr.[1)]	Access Mode		Reset Class	Description See
			Read	Write		
DMA_MExCHCR	DMA Move Engine x Channel Control Register (x = 0-1)	0194_H + (x * 1000_H)	U, SV	BE	3	Page 15-112
DMA_MExSHADR	DMA Move Engine x Channel Shadow Address Register (x = 0-1)	0198_H + (x * 1000_H)	U, SV	BE	3	Page 15-111
DMA_MExCHSR	DMA Move Engine x Channel Status Register (x = 0-1)	$019C_H$ + (x * 1000_H)	U, SV	BE	3	Page 15-108
-	Reserved (x = 0-1)	$01A0_H$ + (x * 1000_H) - $11FC_H$	BE	BE	-	-
DMA_OTSS	DMA OCDS Trigger Set Select Register	1200_H	U, SV	SV	1	Page 15-130
DMA_ERRINTR	DMA Error Interrupt Register	1204_H	U, SV	SV	3	Page 15-133
DMA_PRR0	DMA Pattern Read Register 0	1208_H	U, SV	SV	3	Page 15-134
DMA_PRR1	DMA Pattern Read Register 1	$120C_H$	U, SV	SV	3	Page 15-135
DMA_TIME	DMA Time Register	1210_H	U, SV	BE	3	Page 15-136
-	Reserved	1214_H - $12FC_H$	BE	BE	-	-
DMA_MODEy	DMA Hardware Resource y Mode Register (y = 0-3)	1300_H + (y * 4_H)	U, SV	SV, SE, P00, P01	3	Page 15-137
-	Reserved	1310_H - $17FC_H$	BE	BE	-	-

Table 15-12 Registers Overview - DMA Registers

Short Name	Description	Offset Addr.[1)]	Access Mode		Reset Class	Description See
			Read	Write		
DMA_HRRz	DMA Channel Hardware Resource Register z (z = 000-015)	1800_H + (z * 0004_H)	U, SV	SV, SE, P00, P01	3	Page 15-138
-	Reserved (ch = 16)	1800_H + (ch * 0004_H) - $19FC_H$	BE	BE	-	-
DMA_SUSENRz	DMA Channel Suspend Enable Register z (z = 000-015)	$1A00_H$ + (z * 0004_H)	U, SV	SV, E, Py	1	Page 15-139
-	Reserved (ch = 16)	$1A00_H$ + (ch * 0004_H) - $1BFC_H$	BE	BE	-	-
DMA_SUSACRz	DMA Channel Suspend Acknowledge Register z (z = 000-015)	$1C00_H$ + (z * 0004_H)	U, SV	BE	1	Page 15-140
-	Reserved (ch = 16)	$1C00_H$ + (ch * 0004_H) - $1DFC_H$	BE	BE	-	-
DMA_TSRz	DMA Channel Transaction State Register z (z = 000-015)	$1E00_H$ + (z * 0004_H)	U, SV	SV, Py	3	Page 15-141
-	Reserved (ch = 16)	$1E00_H$ + (ch * 0004_H) - $1FFC_H$	BE	BE	-	-
DMA_RDCRCz	DMA Channel z Read Data CRC (z = 000-015)	2000_H + (z * 0020_H)	U, SV	SV, Py	-	Page 15-144

Table 15-12 Registers Overview - DMA Registers

Short Name	Description	Offset Addr.[1]	Access Mode		Reset Class	Description See
			Read	Write		
DMA_SDCRCz	DMA Channel z Source & Destination Address CRC (z = 000-015)	2004_H + (z * 0020_H)	U, SV	SV, Py	-	Page 15-145
DMA_SADRz	DMA Channel z Source Address (z = 000-015)	2008_H + (z * 0020_H)	U, SV	SV, Py	-	Page 15-146
DMA_DADRz	DMA Channel z Destination Address (z = 000-015)	$200C_H$ + (z * 0020_H)	U, SV	SV, Py	-	Page 15-147
DMA_ADICRz	DMA Channel z Address and Interrupt Control (z = 000-015)	2010_H + (z * 0020_H)	U, SV	SV, Py	-	Page 15-148
DMA_CHCFGRz	DMA Channel z Configuration (z = 000-015)	2014_H + (z * 0020_H)	U, SV	SV, Py	-	Page 15-150
DMA_SHADRz	DMA Channel z Shadow Address (z = 000-015)	2018_H + (z * 0020_H)	U, SV	SV, Py[3]	-	Page 15-152
DMA_CHCSRz	DMA Channel z Control and Status (z = 000-015)	$201C_H$ + (z * 0020_H)	U, SV	SV, Py	-	Page 15-153
-	Reserved (ch = 16)	2000_H + (ch * 0020_H) - $3FFC_H$	BE	BE	-	-

1) The absolute register address is calculated as follows:
Module Base Address (**Table 15-11**) + Offset Address (shown in this column)
Further, the following ranges for parameters x, y, and z are valid: x = 0-1, y = 0-3, z = 000-015.

2) Registers (addresses) protected by the default Hardware Resource Partition 0 Access Enable Registers ACCEN00 and ACCEN01 are marked P00 and P01.

3) In shadow address modes write access to DMA_SHADRz is controlled by the register bit DMA_ADICRz.SHCT[2]. If DMA_ADICRz.SHCT =´0001´ or ´0010´ then Access Mode Write for DMA_SHADRz is BE. If DMA_ADICRz.SHCT =´0101´ or ´0110´ then Access Mode Write for DMA_SHADRz is SV.

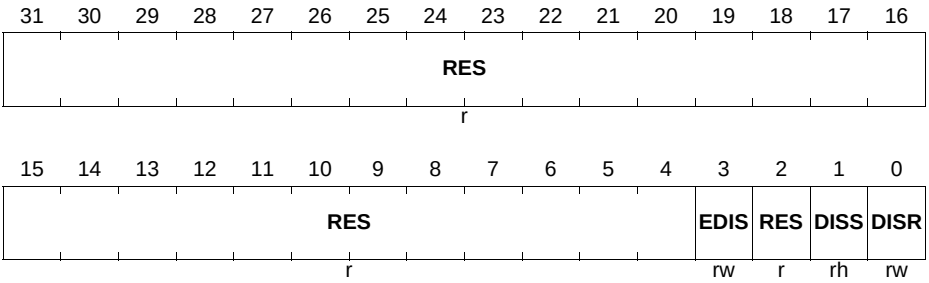
15.8.1 DMA General Module Control Registers

DMA Clock Control Register

The Clock Control Registers DMA_CLC controls the internal f_{DMA} clock signal and sleep mode in order to control the power consumption.

DMA_CLC
DMA Clock Control Register

(0000_H) **Reset Value: $0000\ 0008_H$**



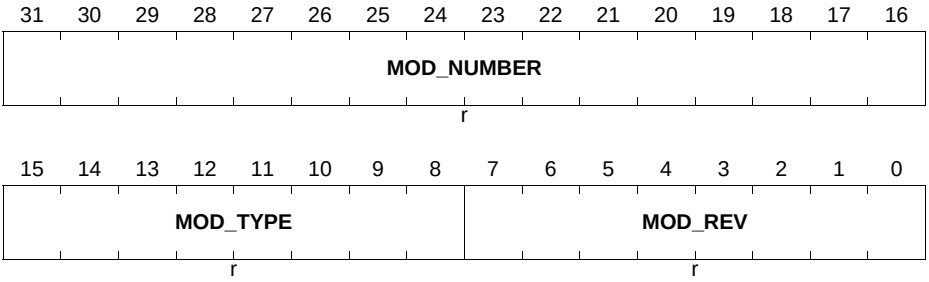
Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module
EDIS	3	rw	**Sleep Mode Enable Control** Used for module sleep mode control. 0_B Sleep mode request is regarded. Module is enabled to go into sleep mode. 1_B Sleep mode request is disregarded. Sleep mode cannot be entered on a request.
RES	2, [31:4]	r	**Reserved** Read as 0; must be written with 0.

Note: After a hardware reset operation, the DMA module is enabled.

Note: The sleep mode does not modify any of the registers.

DMA Module Identification Register

DMA_ID
Module Identification Register (0008$_H$) Reset Value: 0087 C003$_H$

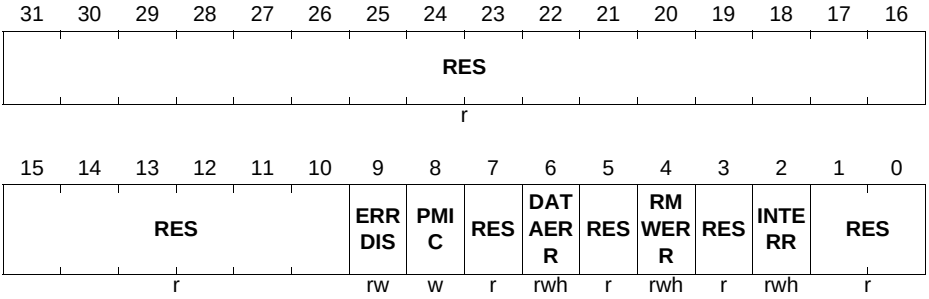


Field	Bits	Type	Description
MOD_REV	[7:0]	r	**Module Revision Number** This bit field defines the module revision number.
MOD_TYPE	[15:8]	r	**Module Type** The bit field is set to C0$_H$ which defines the module as a 32-bit module.
MOD_NUMBER	[31:16]	r	**Module Number Value** This bit field defines a module identification number. The value for the DMA module is 0087$_H$.

DMA Memory Control Register

Provides control of the memory integrity error checking, error signalling to the SMU and error injection for ECC logic test. The register is cleared by an Application Reset.

DMA_MEMCON
DMA Memory Control Register (0020$_H$) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
INTERR	2	rwh	**Internal ECC Error** Flag set by hardware when the DMA logic detects an ECC error while accessing the RAM. This bit is cleared by writing 0_B but cannot be set by software. 0_B No error has occurred 1_B An error has been observed during a RAM access.
RMWERR	4	rwh	**Internal Read Modify Write Error** Flag set by hardware when the DMA logic detects an ECC error on the read phase of an internal RMW operation. This bit is cleared by writing 0_B but cannot be set by software. 0_B No error has occurred 1_B An error has been observed during an internal RMW operation.

Field	Bits	Type	Description
DATAERR	6	rwh	**SPB Data Phase ECC Error** Flag set by hardware when the SPB interface detects an ECC error during the data phase of an incoming read transaction. This bit is cleared by writing 0_B but cannot be set by software. 0_B No error has occurred 1_B An ECC error has been observed on an SPBI transaction addressed to the DMARAM
PMIC	8	w	**Protection Bit for Memory Integrity Control Bit** Will always return 0_B when read 0_B Bit Protection: Bit 9 remains unchanged after DMA_MEMCON write. 1_B Bit 9 will be updated by the current write to DMA_MEMCON
ERRDIS	9	rw	**ECC Error Disable** When set SPB bus errors caused by ECC errors in data read from the SRAM will be disabled 0_B Normal behavior. SPB bus error will occur on SRAM ECC errors. Default after reset 1_B Test Mode. SPB bus errors will not be generated on an SRAM ECC error. This does not affect the generation of interrupts.
RES	[1:0],3,5,7,[31:10]	r	**Reserved** **Read as 0; must be written with 0.**

DMARAM Physical Implementation

If the number of DMA channels is small then the DMARAM may be implemented using logic. A logic implementation shall not generate ECC errors (MEMCON.INTERR = 0_B, MEMCON.RMWERR = 0_B and MEMCON.DATAERR = 0_B).

15.8.2 DMA Access Protection Registers

Write access to each DMA channel z transaction control set is controlled by the respective pair of Access Control Registers for the Hardware Resource assigned to the DMA channel z.

DMA Hardware Resource y Access Enable Register 0

The DMA Access Enable Register 0 controls read / write access for transactions with the on chip bus master TAG ID 000000_B to 011111_B. Each register bit enables one possible 6 bit TAG ID coding.

Mapping of TAG IDs to DMA_ACCEN0.ENn:

- EN0 -> TAG ID 000000_B
- EN1 -> TAG ID 000001_B
- ...
- EN31 -> TAG ID 011111_B

DMA_ACCENy0 (y = 0-3)
DMA Hardware Resource y Access Enable Register 0
(0040_H + y * 8_H) **Reset Value: FFFF FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n = 0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables read / write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

The DMA Hardware Resource y Access Enable Register 0 is safe endinit protected.

DMA Hardware Resource y Access Enable Register 1

The DMA Access Enable Register y1 controls read / write access for transactions with the on chip bus master TAG ID 100000_B to 111111_B. Each register bit enables one possible 6 bit TAG ID coding.

Mapping of TAG IDs to DMA_ACCENx1.ENn:

- EN0 -> TAG ID 100000_B
- EN1 -> TAG ID 100001_B
- ...
- EN31 -> TAG ID111111_B

DMA_ACCENy1 (y = 0-3)
DMA Hardware Resource y Access Enable Register 1

(0044_H + y * 8_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES															
r															

Field	Bits	Type	Description
RES	[31:0]	r	**Reserved** Read as 0; must be written with 0.

The DMA Hardware Resource y Access Enable Register 1 is safe endinit protected.

15.8.3 DMA Sub-block Error Registers

DMA Enable Error Register

The Enable Error Register describes how the DMA controller reacts to errors. It enables the interrupts for Move Engine errors.

DMA_EERx (x = 0-1)
DMA Enable Error Register x

(0120_H + x * 1000_H) **Reset Value: $0503\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES					ELER	RES	ERER	RES						EDER	ESER
r					rw	r	rw	r						rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES															
r															

Field	Bits	Type	Description
RES	[15:0]	r	**Reserved** Read as 0; must be written with 0.
ESER	16	rw	**Enable Move Engine x Source Error** This bit enables the generation of a Move Engine x source error interrupt. 0_B Move Engine x source error interrupt is disabled. 1_B Move Engine x source error interrupt is enabled.
EDER	17	rw	**Enable Move Engine x Destination Error** This bit enables the generation of a Move Engine x destination error interrupt. 0_B Move Engine x destination error interrupt is disabled. 1_B Move Engine x destination error interrupt is enabled.
RES	[23:18]	r	**Reserved** Read as 0; must be written with 0.

Field	Bits	Type	Description
ERER	24	rw	**Enable Move Engine x RAM Error** This bit enables the generation of a Move Engine x RAM error interrupt. 0_B Move Engine x RAM error interrupt is disabled. 1_B Move Engine x RAM error interrupt is enabled.
RES	25	r	**Reserved** Read as 0; must be written with 0.
ELER	26	rw	**Enable Move Engine x DMA Linked List Error** This bit enables the generation of a Move Engine x DMA Linked List error interrupt. 0_B Move Engine x DMA Linked List error interrupt is disabled. 1_B Move Engine x DMA Linked List error interrupt is enabled.
RES	[31:27]	r	**Reserved** Read as 0; must be written with 0.

DMA Error Status Register

The Error Status Register indicates if the DMA controller could not answer to a request because the previous request was not terminated (see **Section 15.4.4.1**).

It indicates that an SPB Bus access or an SRI Bus access has terminated with errors.

DMA_ERRSRx (x = 0-1)
DMA Error Status Register x

(0124_H + x * 1000_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES					DLLER	SLLER	RAMER	RES		SRIER	SPBER	RES		DER	SER
r					rh	rh	rh	r		rh	rh	r		rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES									LEC						
r									rh						

Field	Bits	Type	Description
LEC	[6:0]	rh	**Move Engine x Last Error Channel** This bit field indicates the channel number of the last channel of Move Engine x leading to an On Chip Bus error that has occurred. *Note: Last Error Channel DMA_ERRSRx.LEC is not set on RAM error DMA_ERRSRx.RAMER or Safe Linked List error DMA_ERRSRx.SLLER*
RES	[15:7]	r	**Reserved** Read as 0; must be written with 0.
SER	16	rh	**Move Engine x Source Error** This bit is set whenever a Move Engine x error occurred during a source (read) move of a DMA transfer, or a request could not been serviced due to the access protection. 0_B No Move Engine x source error has occurred. 1_B A Move Engine x source error has occurred.

Field	Bits	Type	Description
DER	17	rh	**Move Engine x Destination Error** This bit is set whenever a Move Engine x error occurred during a destination (write) move of a DMA transfer, or a request could not been serviced due to the access protection. 0_B No Move Engine x destination error has occurred. 1_B A Move Engine x destination error has occurred.
RES	[19:18]	r	**Reserved** Read as 0; must be written with 0.
SPBER	20	rh	**Move Engine x SPB Bus Error** This bit is set whenever a Move Engine x DMA Move that has been started by the DMA SPB master interface leads to an error on the SPB Bus. 0_B No error occurred. 1_B An error occurred on SPB Bus interface.
SRIER	21	rh	**Move Engine x SRI Bus Error** This bit is set whenever a Move Engine x DMA Move that has been started by the DMA SRI master interface leads to an error on the SRI Bus. 0_B No error occurred. 1_B An error occurred on SRI Bus interface.
RES	[23:22]	r	**Reserved** Read as 0; must be written with 0.
RAMER	24	rh	**Move Engine x RAM Error** This bit is set whenever a Move Engine x error occurred during the loading of a transaction control set from the DMARAM to the DMA sub-block x channel registers. 0_B No error occurred. 1_B An error occurred during the load of a transaction control set.
SLLER	25	rh	**Move Engine x Safe Linked List Error** This bit is set whenever a Move Engine x error occurred during the comparison of a SDCRC checksums during a safe linked list. 0_B No error occurred. 1_B An error occurred during the SDCRC checksum comparison.

Field	Bits	Type	Description
DLLER	26	rh	**Move Engine x DMA Linked List Error** This bit is set whenever a Move Engine x error occurred during the loading of a new transaction control set from anywhere in memory to overwrite the current transaction control set stored in the DMARAM. 0_B No error occurred. 1_B An error occurred during the loading of a new transaction control set. *Note: Error reporting includes all DMA Linked List operations included Accumulated Linked List, Safe Linked List and Conditional Linked List.*
RES	[31:27]	r	**Reserved** Read as 0; must be written with 0.

DMA Clear Error Register

The Clear Error contains bits to clear the corresponding Move Engine error flags.

DMA_CLREx (x = 0-1)
DMA Clear Error Register x

(0128_H + x * 1000_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES					CDLLER	CSLLER	CRAMER	RES		CSRIER	CSPBER	RES		CDER	CSER
r					w	w	w	r		w	w	r		w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES															
r															

Field	Bits	Type	Description
RES	[15:0]	r	**Reserved** Read as 0; must be written with 0.
CSER	16	w	**Clear Move Engine x Source Error** 0_B No action 1_B Clear source error flag DMA_ERRSRx.SER.
CDER	17	w	**Clear Move Engine x Destination Error** 0_B No action 1_B Clear destination error flag DMA_ERRSRx.DER.
RES	[19:18]	r	**Reserved** Read as 0; must be written with 0.
CSPBER	20	w	**Clear SPB Error** 0_B No action 1_B Clear error flag DMA_ERRSRx.SPBER.
CSRIER	21	w	**Clear SRI Error** 0_B No action 1_B Clear error flag DMA_ERRSRx.SRIER.
RES	[23:22]	r	**Reserved** Read as 0; must be written with 0.

Field	Bits	Type	Description
CRAMER	24	w	**Clear RAM Error** 0_B No action 1_B Clear error flag DMA_ERRSRx.RAMER.
CSLLER	25	w	**Clear SLL Error** 0_B No action 1_B Clear error flag DMA_ERRSRx.SLLER.
CDLLER	26	w	**Clear DLL Error** 0_B No action 1_B Clear error flag DMA_ERRSRx.DLLER.
RES	[31:27]	r	**Reserved** Read as 0; must be written with 0.

15.8.4 DMA Sub-block Move Engine Registers

DMA Move Engine Status Register

The Move Engine Status Register is a read-only register that holds status information about the transaction handled by the Move Engines.

DMA_MExSR (x = 0-1)
DMA Move Engine x Status Register

(0130_H + x * 1000_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES									CH						
r									rh						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES											WS	RES			RS
r											rh	r			rh

Field	Bits	Type	Description
RS	0	rh	**Move Engine x Read Status** 0_B Move Engine x is not performing a read. 1_B Move Engine x is performing a read.
RES	[3:1]	r	**Reserved** Read as 0; must be written with 0.
WS	4	rh	**Move Engine x Write Status** 0_B Move Engine x is not performing a write. 1_B Move Engine x is performing a write.
RES	[15:5]	r	**Reserved** Read as 0; must be written with 0.
CH	[22:16]	rh	**Active Channel z in Move Engine x** This bit field indicates the number of the DMA Channel z currently being processed by the Move Engine x.
RES	[31:23]	r	**Reserved** Read as 0; must be written with 0.

DMA Move Engine x Read Register 0

The Move Engine x Read Register indicate the value that has just been read by Move Engine x.

The value in the Read Register is compared to the bits in Move Engine Pattern Register depending on the pattern detection configuration and the channel data width.

DMA_MEx0R (x = 0-1)
DMA Move Engine x Read Register 0

(0140_H + x * 1000_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RD03								RD02							
rh								rh							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RD01								RD00							
rh								rh							

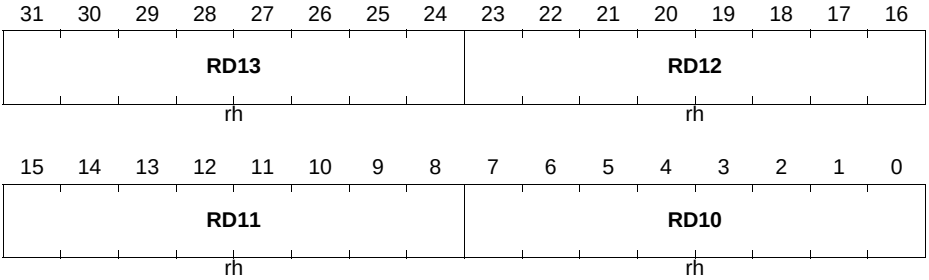
Field	Bits	Type	Description
RD00, RD01, RD02, RD03	[7:0], [15:8], [23:16], [31:24]	rh	**Read Value for Move Engine x** Contains the 32-bit read data (four bytes RD0[3:0]) that is stored in the Move Engine x after each read move. The content of MExR is overwritten after each read move of a DMA channel belonging to DMA Sub-block x.

DMA Move Engine x Read Register 1

The Move Engine x Read Register indicate the value that has just been read by Move Engine x.

DMA_MEx1R (x = 0-1)
DMA Move Engine x Read Register 1

(0144_H + x * 1000_H) **Reset Value: 0000 0000_H**



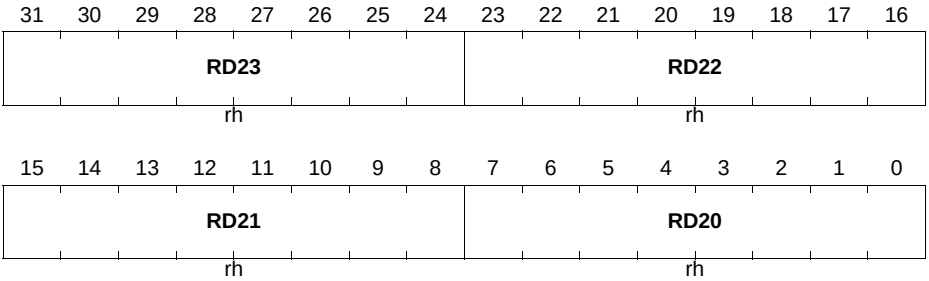
Field	Bits	Type	Description
RD10, RD11, RD12, RD13	[7:0], [15:8], [23:16], [31:24]	rh	**Read Value for Move Engine x** Contains the 32-bit read data (four bytes RD1[3:0]) that is stored in the Move Engine x after each read move. The content of MExR is overwritten after each read move of a DMA channel belonging to DMA Sub-block x.

DMA Move Engine x Read Register 2

The Move Engine x Read Register indicate the value that has just been read by Move Engine x.

DMA_MEx2R (x = 0-1)
DMA Move Engine x Read Register 2

(0148_H + x * 1000_H) **Reset Value: 0000 0000_H**



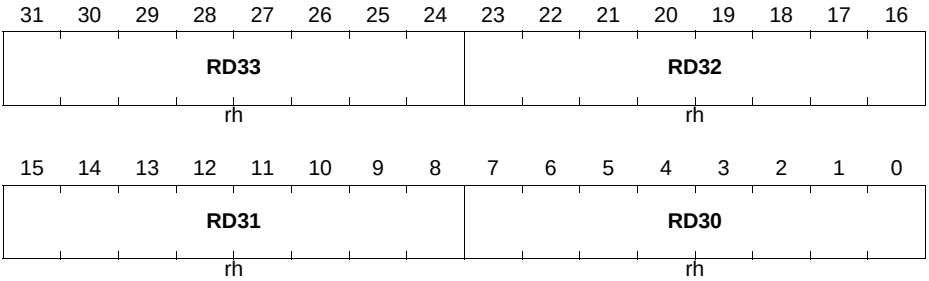
Field	Bits	Type	Description
RD20, RD21, RD22, RD23	[7:0], [15:8], [23:16], [31:24]	rh	**Read Value for Move Engine x** Contains the 32-bit read data (four bytes RD2[3:0]) that is stored in the Move Engine x after each read move. The content of MExR is overwritten after each read move of a DMA channel belonging to DMA sub-block x.

DMA Move Engine x Read Register 3

The Move Engine x Read Register indicate the value that has just been read by Move Engine x.

DMA_MEx3R (x = 0-1)
DMA Move Engine x Read Register 3

($014C_H$ + x * 1000_H) **Reset Value: 0000 0000_H**



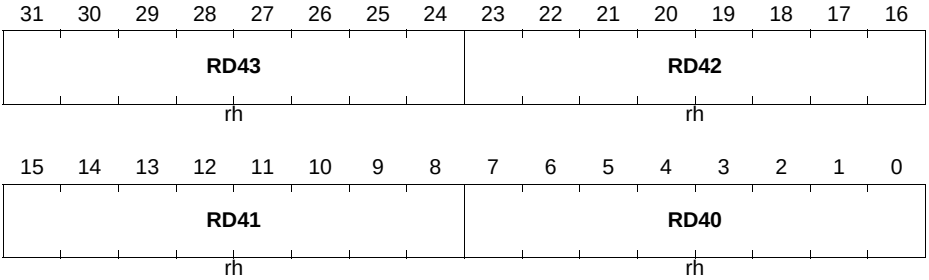
Field	Bits	Type	Description
RD30, RD31, RD32, RD33	[7:0], [15:8], [23:16], [31:24]	rh	**Read Value for Move Engine x** Contains the 32-bit read data (four bytes RD3[3:0]) that is stored in the Move Engine x after each read move. The content of MExR is overwritten after each read move of a DMA channel belonging to DMA sub-block x.

DMA Move Engine x Read Register 4

The Move Engine x Read Register indicate the value that has just been read by Move Engine x.

DMA_MEx4R (x = 0-1)
DMA Move Engine x Read Register 4

(0150_H + x * 1000_H) **Reset Value: 0000 0000_H**



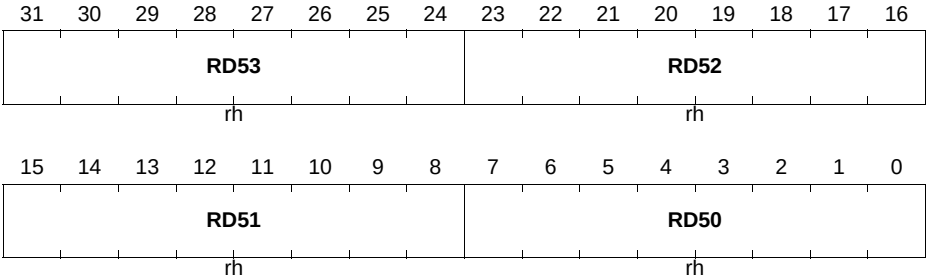
Field	Bits	Type	Description
RD40, RD41, RD42, RD43	[7:0], [15:8], [23:16], [31:24]	rh	**Read Value for Move Engine x** Contains the 32-bit read data (four bytes RD4[3:0]) that is stored in the Move Engine x after each read move. The content of MExR is overwritten after each read move of a DMA channel belonging to DMA sub-block x.

DMA Move Engine x Read Register 5

The Move Engine x Read Register indicate the value that has just been read by Move Engine x.

DMA_MEx5R (x = 0-1)
DMA Move Engine x Read Register 5

(0154_H + x * 1000_H) **Reset Value: 0000 0000_H**



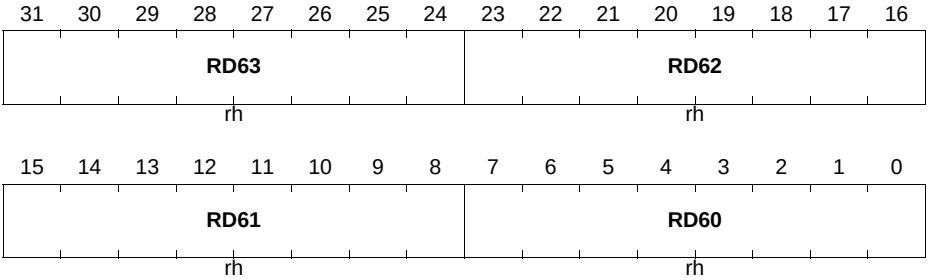
Field	Bits	Type	Description
RD50, RD51, RD52, RD53	[7:0], [15:8], [23:16], [31:24]	rh	**Read Value for Move Engine x** Contains the 32-bit read data (four bytes RD5[3:0]) that is stored in the Move Engine x after each read move. The content of MExR is overwritten after each read move of a DMA channel belonging to DMA sub-block x.

DMA Move Engine x Read Register 6

The Move Engine x Read Register indicate the value that has just been read by Move Engine x.

DMA_MEx6R (x = 0-1)
DMA Move Engine x Read Register 6

(0158_H + x * 1000_H) **Reset Value: 0000 0000_H**



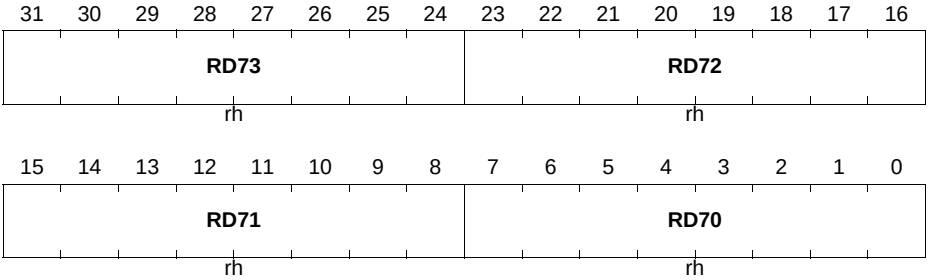
Field	Bits	Type	Description
RD60, RD61, RD62, RD63	[7:0], [15:8], [23:16], [31:24]	rh	**Read Value for Move Engine x** Contains the 32-bit read data (four bytes RD6[3:0]) that is stored in the Move Engine x after each read move. The content of MExR is overwritten after each read move of a DMA channel belonging to DMA sub-block x.

DMA Move Engine x Read Register 7

The Move Engine x Read Register indicate the value that has just been read by Move Engine x.

DMA_MEx7R (x = 0-1)
DMA Move Engine x Read Register 7

($015C_H$ + x * 1000_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
RD70, RD71, RD72, RD73	[7:0], [15:8], [23:16], [31:24]	rh	**Read Value for Move Engine x** Contains the 32-bit read data (four bytes RD7[3:0]) that is stored in the Move Engine x after each read move. The content of MExR is overwritten after each read move of a DMA channel belonging to DMA sub-block x.

15.8.5 DMA Move Engine Active Channel Registers

DMA Move Engine Channel Status Register

The read only DMA active channel Channel Status Register contains the current transfer count, pattern detection compare result and the status of traffic management interrupt triggers.

DMA_MExCHSR (x = 0-1)
DMA Move Engine x Channel Status Register
($019C_H$ + x * 1000_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES								FROZEN	BUFFER	RES		IPM	ICH	WRPD	WRPS
r								rh	rh	r		rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LXO	RES	TCOUNT													
rh	r	rh													

Field	Bits	Type	Description
TCOUNT	[13:0]	rh	**Transfer Count Status** TCOUNT holds the actual value of the DMA transfer count for DMA channel z. TCOUNT is loaded with the value of CHCFGRz.TREL when TSRz.CH becomes set (and TCOUNT = 0). After each DMA transfer, TCOUNT is decremented by 1.
RES	14	r	**Reserved** Read as 0; must be written with 0.

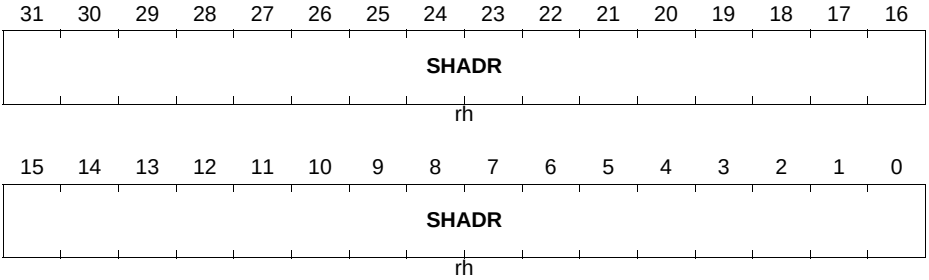
Field	Bits	Type	Description
LXO	15	rh	**Old Value of Pattern Detection** This bit contains the compare result of a pattern compare operation when 8-bit or 16-bit data width is selected. 8-bit data width: see **Table 15-4** and **Figure 15-27** 16-bit data width: see **Table 15-5** and **Figure 15-28** 0_B The corresponding pattern compare operation did not find a pattern match during the previous DMA read move 1_B The corresponding pattern compare operation found a pattern match during the previous DMA read move
WRPS	16	rh	**Wrap Source Buffer** Indicates a wrap-around of source buffer. 0_B No wrap-around occurred for channel z. 1_B A wrap-around occurred for channel z. This bit is reset by software by writing a 1 to CHCSRz.CWRP or TSRz.RST.
WRPD	17	rh	**Wrap Destination Buffer** Indicates a wrap-around of destination buffer. 0_B No wrap-around occurred for channel z. 1_B A wrap-around occurred for channel z. This bit is reset by software by writing a 1 to CHCSRz.CWRP or TSRz.RST.
ICH	18	rh	**Interrupt from Channel** This bit indicates that channel z has raised an interrupt for TCOUNT = IRDV or if TCOUNT has been decremented (depending on ADICRz.INTCT[0]. This bit (and IPM) is reset by software when writing a 1 to ADICRz.CICH or by a channel reset (writing TSRz.RST = 1). 0_B A channel interrupt has not been detected. 1_B A channel interrupt has been detected.

Field	Bits	Type	Description
IPM	19	rh	**Pattern Detection from Channel** This bit indicates that a pattern has been detected for channel z while the pattern detection has been enabled. This bit (and ICH) is reset by software when writing a 1 to ADICRz.CICH or by a channel reset (writing TSRz.RST = 1). 0_B A pattern has not been detected. 1_B A pattern has been detected.
RES	[21:20]	r	**Reserved** Read as 0; must be written with 0.
BUFFER	22	rh	**DMA Double Buffering Active Buffer** This bit is active during DMA double buffering and indicates which buffer is read or filled. 0_B Buffer 0 read or filled by DMA. 1_B Buffer 1 read or filled by DMA.
FROZEN	23	rh	**DMA Double Buffering Frozen Buffer** This bit is active during DMA double buffering and indicates that one of the double buffers is frozen and available for a cyclic software task. 0_B Buffer is not frozen. 1_B Buffer is frozen.
RES	[31:24]	r	**Reserved** Read as 0; must be written with 0.

DMA Move Engine Channel Shadow Address Register

The read only DMA active channel Shadow Address Register holds the 32-bit shadow address.

DMA_MExSHADR (x = 0-1)
DMA Move Engine x Channel Shadow Address Register
(0198_H + x * 1000_H) **Reset Value: 0000 0000_H**

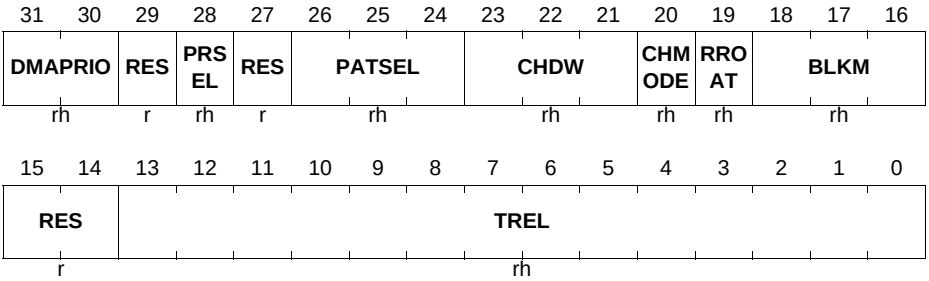


Field	Bits	Type	Description
SHADR	[31:0]	rh	**Shadowed Address** This bit field holds the 32-bit shadow address of the active DMA channel. The function of the shadow address is set by the shadow control settings. See **Table 15-15** for a description.

DMA Move Engine Channel Control Register

The read only DMA active channel Channel Control Register contains the configuration and control bits.

DMA_MExCHCR (x = 0-1)
DMA Move Engine x Channel Control Register
(0194_H + x * 1000_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
TREL	[13:0]	rh	**Transfer Reload Value** This bit field contains the number of DMA transfers for a DMA transaction of DMA channel z. This14-bit transfer count value is loaded into MExCHSR.TCOUNT at the start of a DMA transaction (when TSRz.CH becomes set and CHCSRz.TCOUNT = 0). A write to CHCFGz.TREL during a running DMA transaction has no influence to the running DMA transaction. If CHCFGRz.TREL = 0 or if CHCFGRz.TREL = 1 then MExCHSR.TCOUNT will be loaded with 1 when a new transaction is started (at least one DMA transfer must be executed per DMA transaction).

Field	Bits	Type	Description
BLKM	[18:16]	rh	**Block Mode** BLKM determines the number of DMA moves executed during one DMA transfer. 000_B One DMA transfer has 1 DMA move 001_B One DMA transfer has 2 DMA moves 010_B One DMA transfer has 4 DMA moves 011_B One DMA transfer has 8 DMA moves 100_B One DMA transfer has 16 DMA moves 101_B One DMA transfer has 3 DMA moves 110_B One DMA transfer has 5 DMA moves 111_B One DMA transfer has 9 DMA moves See also **Figure 15-12** on **Page 15-23**.
RROAT	19	rh	**Reset Request Only After Transaction** RROAT determines whether or not the TSRz.CH transfer request state flag is reset after each transfer. 0_B TSRz.CH is reset after each transfer. A transfer request is required for each transfer. 1_B TSRz.CH is reset when CHSRz.TCOUNT = 0 after a transfer. One transfer request starts a complete DMA transaction
CHMODE	20	rh	**Channel Operation Mode** CHMODE determines the reset condition for control bit TSRz.HTRE of the DMA channel. 0_B Single Mode operation is selected for DMA channel. After a transaction, DMA channel is disabled for further hardware requests (TSRz.HTRE is reset by hardware) TSRz.HTRE must be set again by software for starting a new transaction. 1_B Continuous Mode operation is selected for DMA channel z. After a transaction, bit TSRz.HTRE remains set

Field	Bits	Type	Description
CHDW	[23:21]	rh	**Channel Data Width** CHDW determines the data width for the read and write moves of DMA channel z. 000_B 8-bit data width for moves selected *Note: Single Data Transfer Byte (SDTB)* 001_B 16-bit data width for moves selected *Note: Single Data Transfer Half-Word (SDTH)* 010_B 32-bit data width for moves selected *Note: Single Data Transfer Word (SDTW)* 011_B 64-bit data width transaction selected *Note: SRI-Bus: Single Data Transfer Double Word (SDTD)* *Note: SPB-Bus: not supported.* 100_B 128-bit data width transaction selected *Note: SRI-Bus: Block Transfer Request - 2 Transfers (BTR2)* *Note: SPB-Bus: not supported.* 101_B 256-bit data width transaction selected *Note: SRI-Bus: Block Transfer Request - 4 Transfers (BTR4)* *Note: SPB-Bus: not supported.* 110_B Reserved 111_B Reserved

Field	Bits	Type	Description
PATSEL	[26:24]	rh	**Pattern Select** This bit field selects the mode of the pattern detection logic. Depending on the channel data width, PATSEL[1:0] selects different pattern detection configurations. If pattern detection is enabled (PATSEL[1:0] not equal 00_B), the pattern detection interrupt line will be activated on the selected pattern match. PATSEL[2] selects the pattern read register. 0_B Pattern Read Register 0 1_B Pattern Read Register 1 **8-bit channel data width (CHDW = 000_B):** Selected pattern detection configuration see **Table 15-4** on **Page 15-55**. **16-bit channel data width (CHDW = 001_B):** Selected pattern detection configuration see **Table 15-5** on **Page 15-57**. **32-bit channel data width (CHDW = 010_B):** Selected pattern detection configuration see **Table 15-6** on **Page 15-60**.
PRSEL	28	rh	**Peripheral Request Select** This bit field controls the hardware request input multiplexer of DMA channel z-1 (see **Figure 15-6** on **Page 15-13**). 0_B Hardware trigger selected 1_B Daisy chain selected
DMAPRIO	[31:30]	rh	**DMA Priority** This bit determines the DMA the request priority that is used when a move operation related to channel z is requesting SPB-Bus. This bit has no effect in channel prioritization inside the Move Engine x in. 00_B Low priority selected 01_B Medium priority selected 10_B Medium priority selected 11_B High priority selected
RES	[15:14], 27, 29	r	**Reserved** Read as 0; must be written with 0.

DMA Move Engine Channel Address and Interrupt Control Register

The read only DMA active channel Address and Interrupt Control Register controls how source and destination addresses are updated after a DMA move. Furthermore, it determines whether or not a source or destination address register update is shadowed.

The interrupt control bits enable the generation of DMA traffic interrupt triggers.

DMA_MExADICR (x = 0-1)
DMA Move Engine x Channel Address and Interrupt Control Register
(0190_H + x * 1000_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IRDV				INTCT		WRPDE	WRPSE	ETRL	STAMP	DCBE	SCBE	SHCT			
rh				rh		rh	rh	rh	rh	rh	rh	rh			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CBLD				CBLS				INCD	DMF			INCS	SMF		
rh				rh				rh	rh			rh	rh		

Field	Bits	Type	Description
SMF	[2:0]	rh	**Source Address Modification Factor** This bit field and the data width as defined in MExCHCR.CHDW determine an address offset value by which the source address is modified after each DMA move. See also **Table 15-13**. 000_B Address offset is 1 x MExCHCR.CHDW 001_B Address offset is 2 x MExCHCR.CHDW 010_B Address offset is 4 x MExCHCR.CHDW 011_B Address offset is 8 x MExCHCR.CHDW 100_B Address offset is 16 x MExCHCR.CHDW 101_B Address offset is 32 x MExCHCR.CHDW 110_B Address offset is 64 x MExCHCR.CHDW 111_B Address offset is 128 x MExCHCR.CHDW

Field	Bits	Type	Description
INCS	3	rh	**Increment of Source Address** This bit determines whether the address offset as selected by SMF will be added to or subtracted from the source address after each DMA move. The source address is not modified if CBLS = 0000_B. 0_B Address offset will be subtracted 1_B Address offset will be added.
DMF	[6:4]	rh	**Destination Address Modification Factor** This bit field and the data width as defined in MExCHCR.CHDW determines an address offset value by which the destination address is modified after each DMA move. The destination address is not modified if CBLD = 0000_B. See also **Table 15-13**. 000_B Address offset is 1 x MExCHCR.CHDW 001_B Address offset is 2 x MExCHCR.CHDW 010_B Address offset is 4 x MExCHCR.CHDW 011_B Address offset is 8 x MExCHCR.CHDW 100_B Address offset is 16 x MExCHCR.CHDW 101_B Address offset is 32 x MExCHCR.CHDW 110_B Address offset is 64 x MExCHCR.CHDW 111_B Address offset is 128 x MExCHCR.CHDW
INCD	7	rh	**Increment of Destination Address** This bit determines whether the address offset as selected by DMF will be added to or subtracted from the destination address after each DMA move. The destination address is not modified if CBLD = 0000_B. 0_B Address offset will be subtracted 1_B Address offset will be added

Field	Bits	Type	Description
CBLS	[11:8]	rh	**Circular Buffer Length Source** This bit field determines which part of the 32-bit source address register remains unchanged and is not updated after a DMA move operation (see also **Section 15.4.3.8**). Therefore, CBLS also determines the size of the circular source buffer. 0000_B Source address SADR[31:0] is not updated 0001_B Source address SADR[31:1] is not updated 0010_B Source address SADR[31:2] is not updated 0011_B Source address SADR[31:3] is not updated $\ldots_B$ ... 1110_B Source address SADR[31:14] is not updated 1111_B Source address SADR[31:15] is not updated
CBLD	[15:12]	rh	**Circular Buffer Length Destination** This bit field determines which part of the 32-bit destination address register remains unchanged and is not updated after a DMA move operation (see also **Page 15-25**). Therefore, CBLD also determines the size of the circular destination buffer. 0000_B Destination address DADR[31:0] is not updated 0001_B Destination address DADR[31:1] is not updated 0010_B Destination address DADR[31:2] is not updated 0011_B Destination address DADR[31:3] is not updated $\ldots_B$ ... 1110_B Destination address DADR[31:14] is not updated 1111_B Destination address DADR[31:15] is not updated
SHCT	[19:16]	rh	**Shadow Control** This bit field determines the function of the shadow address register. See **Table 15-15**.
SCBE	20	rh	**Source Circular Buffer Enable** 0_B Source circular buffer disabled 1_B Source circular buffer enabled

Field	Bits	Type	Description
DCBE	21	rh	**Destination Circular Buffer Enable** 0_B Destination circular buffer disabled 1_B Destination circular buffer enabled
STAMP	22	rh	**Time Stamp** This bit enables the appendage of a timestamp after the end of the last DMA Move during a DMA transaction. 0_B No action. 1_B Enable the appendage of a 32-bit timestamp.
ETRL	23	rh	**Enable Transaction Request Lost Interrupt** This bit enables the generation of an interrupt when TSRz.TRLz is set. 0_B The interrupt generation for a request lost event for channel z is disabled. 1_B The interrupt generation for a request lost event for channel z is enabled.
WRPSE	24	rh	**Wrap Source Enable** 0_B Wrap source buffer interrupt trigger disabled 1_B Wrap source buffer interrupt trigger enabled
WRPDE	25	rh	**Wrap Destination Enable** 0_B Wrap destination buffer interrupt trigger disabled 1_B Wrap destination buffer interrupt trigger enabled
INTCT	[27:26]	rh	**Interrupt Control** 00_B No interrupt trigger will be generated on changing the TCOUNT value. The bit CHSRz.ICH is set when TCOUNT equals IRDV. 01_B No interrupt trigger will be generated on changing the TCOUNT value. The bit CHSRz.ICH is set when TCOUNT is decremented 10_B Interrupt trigger is generated and bit CHSRz.ICH is set on changing the TCOUNT value and TCOUNT equals IRDV 11_B Interrupt trigger is generated and bit CHSRz.ICH is set each time TCOUNT is decremented *Note: see **Figure 15-31**.* *Note: If DMA_CHCFGRz.PRSEL = 1_B for the next lower priority channel then the channel transfer trigger interrupt is disabled.*

Field	Bits	Type	Description
IRDV	[31:28]	rh	**Interrupt Raise Detect Value** Defines the Threshold Limit of CHSRz.TCOUNT for which a channel interrupt trigger will be raised.

Address Offset Calculation Tables

Table 15-13 and Table 15-14 show the offset values that are added or subtracted to/from a source or destination address register after a DMA move.

Bit field SMF and bit INCS determine the offset value for the source address.

Bit field DMF and bit INCD determine the offset value for the destination address.

Table 15-13 Address Offset Calculation Table (8-bit, 16-bit and 32-bit)

CHCFGRz.CHDW = 000_B (8-bit Data Width)			CHCFGRz.CHDW = 001_B (16-bit Data Width)			CHCFGRz.CHDW = 010_B (32-bit Data Width)		
SMF DMF	**INCS INCD**	**Address Offset**	**SMF DMF**	**INCS INCD**	**Address Offset**	**SMF DMF**	**INCS INCD**	**Address Offset**
000_B	0	-1	000_B	0	-2	000_B	0	-4
	1	+1		1	+2		1	+4
001_B	0	-2	001_B	0	-4	001_B	0	-8
	1	+2		1	+4		1	+8
010_B	0	-4	010_B	0	-8	010_B	0	-16
	1	+4		1	+8		1	+16
011_B	0	-8	011_B	0	-16	011_B	0	-32
	1	+8		1	+16		1	+32
100_B	0	-16	100_B	0	-32	100_B	0	-64
	1	+16		1	+32		1	+64
101_B	0	-32	101_B	0	-64	101_B	0	-128
	1	+32		1	+64		1	+128
110_B	0	-64	110_B	0	-128	110_B	0	-256
	1	+64		1	+128		1	+256
111_B	0	-128	111_B	0	-256	111_B	0	-512
	1	+128		1	+256		1	+512

Table 15-14 Address Offset Calculation Table (64-bit, 128-bit and 256-bit)

CHCFGRz.CHDW = 011_B (64-bit Data Width)			CHCFGRz.CHDW = 100_B (128-bit Data Width)			CHCFGRz.CHDW = 101_B (256-bit Data Width)		
SMF DMF	**INCS INCD**	**Address Offset**	**SMF DMF**	**INCS INCD**	**Address Offset**	**SMF DMF**	**INCS INCD**	**Address Offset**
000_B	0	-8	000_B	0	-16	000_B	0	-32
	1	+8		1	+16		1	+32
001_B	0	-16	001_B	0	-32	001_B	0	-64
	1	+16		1	+32		1	+64
010_B	0	-32	010_B	0	-64	010_B	0	-128
	1	+32		1	+64		1	+128
011_B	0	-64	011_B	0	-128	011_B	0	-256
	1	+64		1	+128		1	+256
100_B	0	-128	100_B	0	-256	100_B	0	-512
	1	+128		1	+256		1	+512
101_B	0	-256	101_B	0	-512	101_B	0	-1024
	1	+256		1	+512		1	+1024
110_B	0	-512	110_B	0	-1024	110_B	0	-2048
	1	+512		1	+1024		1	+2048
111_B	0	-1024	111_B	0	-2048	111_B	0	-4096
	1	+1024		1	+2048		1	+4096

Shadow Control Table

Table 15-15 defines the functionality of the shadow address register control bits.

Table 15-15 Shadow Control Table

SHCT	Description
0000_B	Source and destination registers written directly. *Note: Shadow address register is not used and set to 00000000_H.*
0001_B	**Source Address Buffering (Read Only)** Shadow address used for source buffering. When writing to SADRz, the address is buffered in SHADRz and transferred to SADRz with the start of the next DMA transaction. *Note: Shadow address register is read only and is automatically set to 00000000_H when the shadow transfer takes place (equal to AudoNG).*
0010_B	**Destination Address Buffering (Read Only)** Shadow address used for destination buffering. When writing to DADRz, the address is buffered in SHADRz and transferred to DADRz with the start of the next DMA transaction. *Note: Shadow address register is read only and is automatically set to 00000000_H when the shadow transfer takes place (equal to AudoNG).*
0011_B	Reserved. *Note: Shadow address register is not used and set to 00000000_H.*
0100_B	Reserved. *Note: Shadow address register is not used and set to 00000000_H.*
0101_B	**Source Address Buffering (Direct Write)** Shadow address used for source buffering. Shadow address register can be read and can be directly written. The value stored in SHADRz is not automatically modified when the shadow transfer takes place.
0110_B	**Destination Address Buffering (Direct Write)** Shadow address used for destination buffering. Shadow address register can be read and can be directly written. The value stored in SHADRz is not automatically modified when the shadow transfer takes place.
0111_B	Reserved. *Note: Shadow address register is not used and set to 00000000_H.*
1000_B	**Double Source Buffering** **Software Switch Only** Shadow address used for double buffering

Table 15-15 Shadow Control Table (cont'd)

SHCT	Description
1001_B	**Double Source Buffering** **Automatic Hardware and Software Switch** Shadow address used for double buffering
1010_B	**Destination Double Destination Buffering** **Software Switch Only** Shadow address used for double buffering
1011_B	**Destination Double Destination Buffering** **Automatic Hardware and Software Switch** Shadow address used for double buffering
1100_B	**DMA Linked List** The DMA controller reads a DMA channel transaction control set and overwrites 8 X words in the corresponding DMARAM channel z. *Note: The SDCRC and RDCRC checksums are unique for each DMA transaction in the DMA Linked List. RDCRCRz and SDCRCRz are overwritten after each DMA transaction in the DMA Linked List.*
1101_B	**Accumulated Linked List** The DMA controller reads a DMA channel transaction control set and overwrites 6 X words in the corresponding DMARAM channel z. *Note: The SDCRC and RDCRC checksums are accumulated across all DMA transactions of the Accumulated Linked List. RDCRCRz and SDCRCRz are not overwritten*
1110_B	**Safe Linked List** The DMA controller reads a DMA channel transaction control set. The Linked List only proceeds with the next DMA transaction if the existing SDCRC checksum matches the expected SDCRC checksum in the loaded from the new DMA transaction control set. *Note: The SDCRC and RDCRC checksums are accumulated across all DMA transactions of the Safe Linked List. RDCRCRz is not overwritten.*
1111_B	**Conditional Linked List** Shadow address register (MExSHADR) and source and destination address CRC register (MExSDCRC) are used as address pointers to a Linked List. The selection of the address pointer is determined by DMA channel pattern detection conditions. *Note: Calculation of SDCRC checksums is not supported.*

Shadow Address used for Source or Destination Buffering

The Shadow Address Register holds the shadowed source or destination address before it is written into the source or destination address register.

Shadow Address used for DMA Double Buffering

The Shadow Address Register holds a second source or destination address used to store data in a second source or destination location.

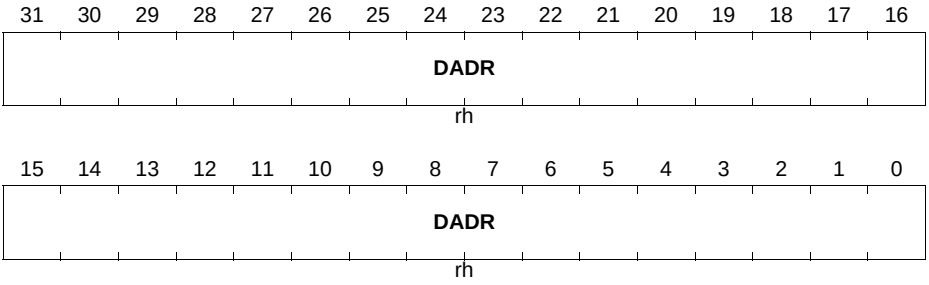
Shadow Address used for DMA Linked Lists

The Shadow Address Register holds an address pointer to the next DMA transaction control set.

DMA Move Engine Channel Destination Address Register

The read only DMA active channel Destination Address Register holds the 32-bit destination address. If a DMA channel is active, MExDADR is updated continuously (if programmed) and shows the actual destination address that is used for write moves within DMA transfers.

DMA_MExDADR (x = 0-1)
DMA Move Engine x Channel Destination Address Register x
($018C_H$ + x * 1000_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
DADR	[31:0]	rh	**Destination Address** This bit field holds the actual 32-bit destination address of the active DMA channel that is used for write moves.

DMA Move Engine x Channel Source Address Register

The read only DMA active channel Source Address Register holds the 32-bit source address. If a DMA channel is active, MExSADR is updated continuously (if programmed) and shows the actual source address that is used for read moves within DMA transfers.

DMA_MExSADR (x = 0-1)
DMA Move Engine x Channel Source Address Register
(0188_H + x * 1000_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SADR															
rh															

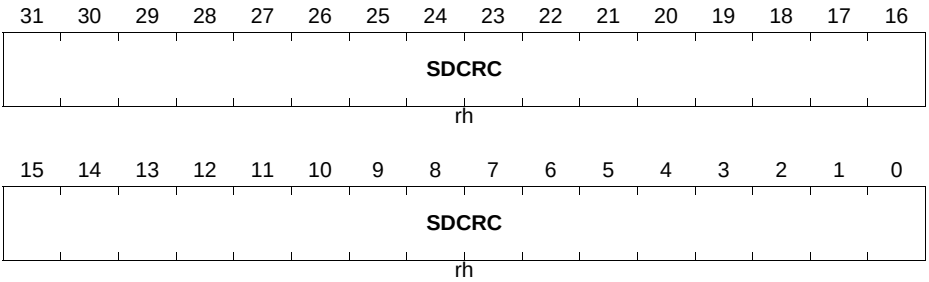
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SADR															
rh															

Field	Bits	Type	Description
SADR	[31:0]	rh	**Source Start Address** This bit field holds the actual 32-bit source address of the active DMA channel that is used for read moves.

DMA Channel Source and Destination Address CRC Register

The read only DMA active channel Source and Destination Address CRC Register will store one polynomial calculation during each DMA read and each DMA write move provided there is no retry or error condition flagged.

DMA_MExSDCRC (x = 0-1)
DMA Move Engine x Channel Source and Destination Address CRC Register
(0184_H + x * 1000_H) **Reset Value: 0000 0000_H**

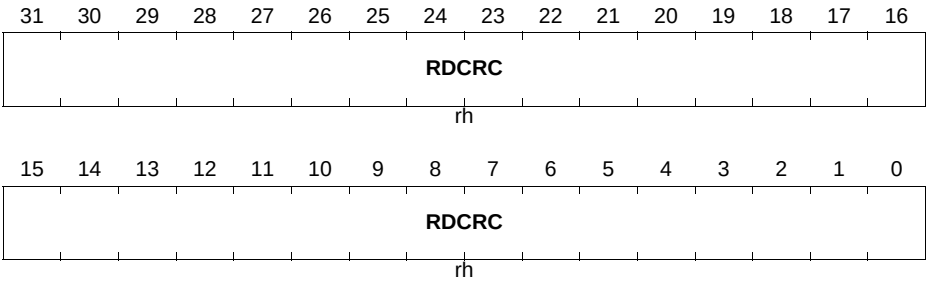


Field	Bits	Type	Description
SDCRC	[31:0]	rh	**Source and Destination Address CRC** This bit field contains the working CRC32 ethernet polynomial checksum for the source and destination address.

DMA Move Engine Channel Read Data CRC Register

The read only DMA active channel Read Data CRC Register will store one polynomial calculation during each DMA read move provided that there is no retry or error condition flagged. In order to start a CRC32 sequence the MExRDCRC must be initialized (e.g. written with 00000000_H or with a desired start value) and an DMA transaction must be set up (start address, length, etc.).

DMA_MExRDCRC (x = 0-1)
DMA Move Engine x Channel Read Data CRC Register
(0180_H + x * 1000_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
RDCRC	[31:0]	rh	**Read Data CRC** This bit field contains the working CRC32 ethernet polynomial checksum for read data.

15.8.6 DMA OCDS Registers

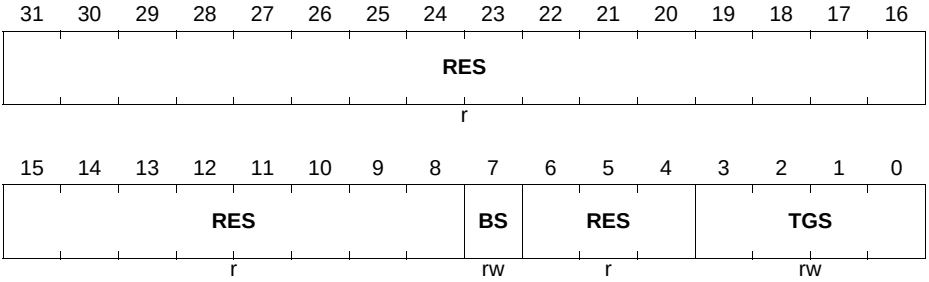
DMA OCDS Trigger Set Select Register

The selection of the OCDS Trigger Bus (OTGB) is controlled by the OTSS register.

The OTSS register is cleared by Debug Reset and it is cleared by each System Reset when OCDS is disabled. The OTSS Register is not touched by the System Reset when OCDS is enabled.

Write access requires OCDS to be enabled and Supervisor mode.

DMA_OTSS
DMA OCDS Trigger Set Select (1200$_H$) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
TGS	[3:0]	rw	**Trigger Set (Table 15-16) for OTGB0/1** 0_D No Trigger Set selected 1_D Trigger Set 1 2_D Trigger Set 2 8_D Trigger Set 8 others reserved
BS	7	rw	**OTGB0/1 Bus Select** 0_B Trigger Set is output on OTGB0 1_B Trigger Set is output on OTGB1
RES	[6:4], [31:8]	r	**Reserved** Read as 0; must be written with 0.

OCDS Trigger Bus Interface

Table 15-16 lists the Trigger Sets, which are selected with the DMA_OTSS register for the 16 bit OCDS Trigger Bus (OTGB0/1) interface. Only one Trigger Set can be output at a time either on OTGB0 or on OTGB1.

The OCDS Trigger Bus has no dependency on the OCDS enabled state.

Table 15-16 DMA Trigger Sets

Index	Description	Details
0	No Trigger Set selected	
1	**Channels (TS16_PF)** Active channels	Table 15-17
2	**Channels (TS16_ERR)** Error channel number and flags	Table 15-18
8	**Transaction Request State (TS16_C15)** Transaction Request State Channel [15:0]	
Other	Reserved. No Trigger Set selected	

Performance Trigger Set

Table 15-17 TS16_PF Trigger Set Channels

Bits	Name	Description
[6:0]	CH0	Channel number active in move engine 0
7	ME0	Move engine 0 idle/active 0_B Move engine 0 is idle 1_B Move engine 0 is active
[14:8]	CH1	Channel number active in move engine 1
15	ME1	Move engine 1 active 0_B Move engine 1 is idle 1_B Move engine 1 is active

The performance trigger set continuously monitors the activity in the move engines. While the move engine is servicing a DMA request from DMA channel z the move engine idle/active bit TS16_PF.MEx is set and the TS16_PF.CHx bit field is set to the DMA channel number z. When the move engine is idle the idle/active bit TS16_PF.MEx goes low. The TS16_PF.CHx bit field retains the value of the last active DMA channel.

Error Trigger Set

Table 15-18 TS16_ERR Trigger Set Channels

Bits	Name	Description
[6:0]	LEC	Last error channel number
[11:7]		Reserved
12	ME0SE	Source Error
13	ME0DE	Destination Error
14	ME1SE	Source Error
15	ME1DE	Destination Error

If a move engine x source or destination error occurs then the respective CH.MExSE or TS16_CH.MExDE gets set. The TS16_CH.LEC returns the last error channel. If both move engines generate an error condition in the same clock cycle then the TS16_CH.LEC will record the last error channel for move engine 0.

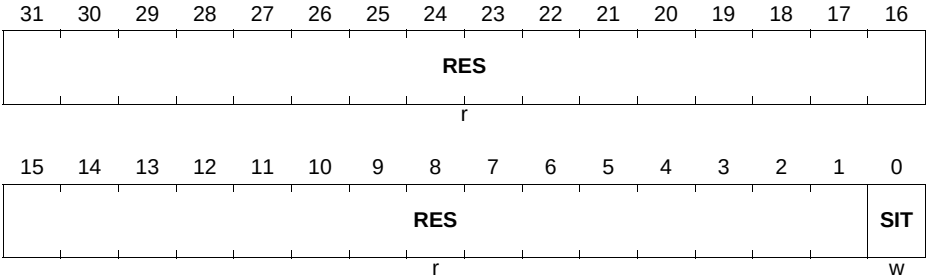
The TS16_ERR trigger set retains its value until there is a new move engine source or destination error. The MCDS must be able to sample the source or destination occurrence and the DMA error channel.

DMA Error Interrupt Set Register

The Error Interrupt Set Register allows the DMA error interrupt service request to be activated by software.

DMA_ERRINTR
DMA Error Interrupt Set Register

(1204_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
SIT	0	w	**Set Error Interrupt Service Request** 0_B No action 1_B DMA error interrupt service request will be activated. Reading this bit returns a 0
RES	[31:1]	r	**Reserved** Read as 0; must be written with 0.

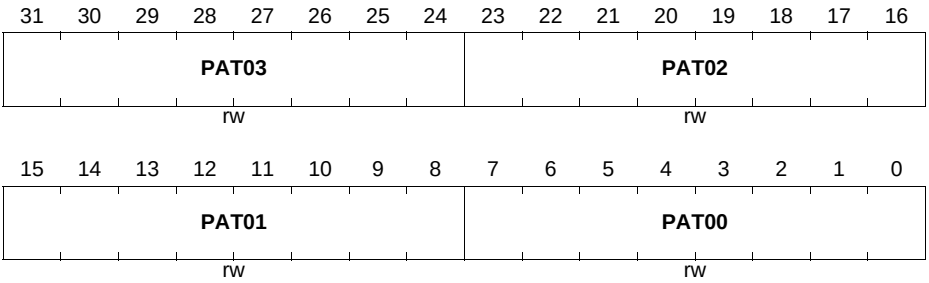
15.8.7 DMA Pattern Detection Registers

Pattern Read Register 0

The Pattern Read Register 0 contains the patterns (mask and/or compare bits) to be processed by the Move Engine pattern detection logic.

DMA_PRR0
Pattern Read Register 0

(1208_H) **Reset Value: 0000 0000_H**



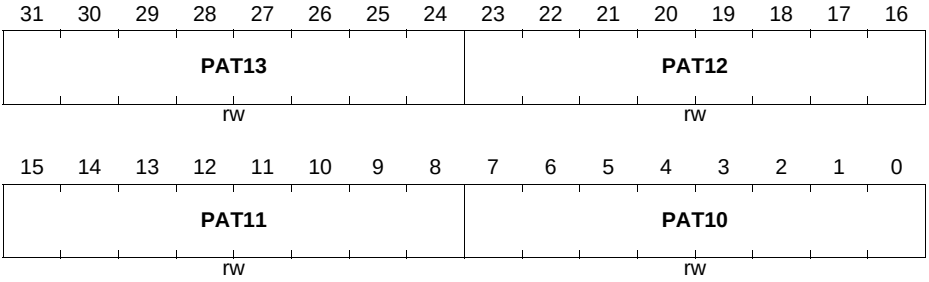
Field	Bits	Type	Description
PAT00, PAT01, PAT02, PAT03	[7:0], [15:8], [23:16], [31:24]	rw	**Pattern for Move Engine** Determines up to four 8-bit compare patterns/mask patterns to be processed by the pattern detection logic in the Move Engine. Depending on the pattern detection configuration (MExCHCR.PATSEL) and channel data width (MExCHCR.CHDW), the patterns are processed as bytes or half-words.

Pattern Read Register 1

The Pattern Read Register 1 contains the patterns (mask and/or compare bits) to be processed by the Move Engine pattern detection logic.

DMA_PRR1
Pattern Read Register 1

($120C_H$) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
PAT10, PAT11, PAT12, PAT13	[7:0], [15:8], [23:16], [31:24]	rw	**Pattern for Move Engine** Determines up to four 8-bit compare patterns/mask patterns to be processed by the pattern detection logic in the Move Engine. Depending on the pattern detection configuration (MExCHCR.PATSEL) and channel data width (MExCHCR.CHDW), the patterns are processed as bytes or half-words.

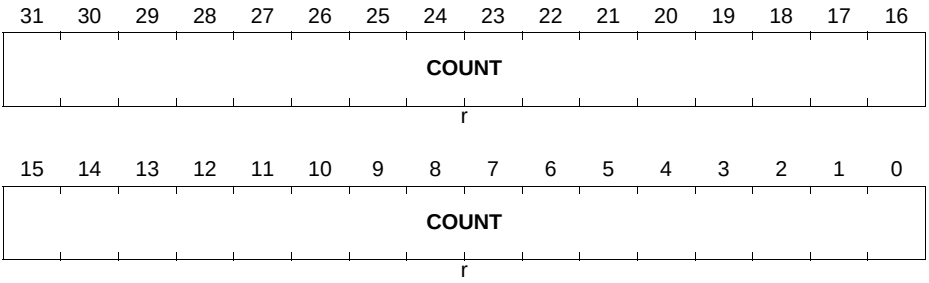
15.8.8 DMA Flow Control Registers

Time Register

The Time Register contains the 32-bit count value used for the appendage of DMA timestamps.

DMA_TIME
Time Register

(1210_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
COUNT	[31:0]	r	**Timestamp Count** This bit field provides the count value used for the appendage of DMA timestamps.

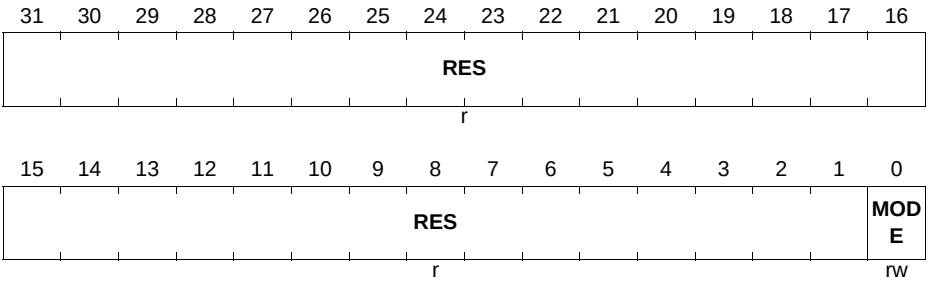
15.8.9 DMA Channel Hardware Resource Registers

DMA Mode Registers

The DMA Mode Registers control whether hardware resources access the on chip buses in supervisor mode or user mode.

DMA_MODEy (y = 0-3)
DMA Mode Register y

(1300_H + y * 4_H) **Reset Value: 0000 0001_H**



Field	Bits	Type	Description
MODE	0	rw	**Hardware Resource Supervisor Mode** Bit determines if a hardware resource 0 makes bus accesses in supervisor mode or user mode. 0_B Bus master is in user mode. 1_B Bus master is in supervisor mode.
RES	[31:1]	r	**Reserved** Read as 0; must be written with 0.

DMA Channel Hardware Resource Register z

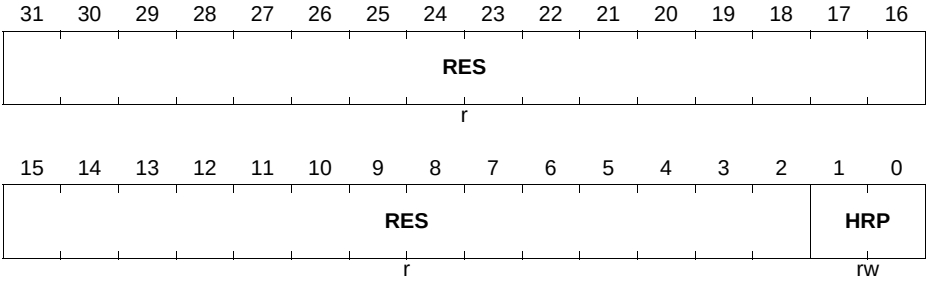
The Hardware Resource Register assigns DMA channels to one of the four hardware resource partitions.

Initially all DMA channels are assigned to Hardware Resource 0 and read/write accesses are controlled by the ACCEN00 and ACCEN01 registers. If a DMA channel z is assigned to one of the other Hardware Resource Partitions then access control will change to the corresponding pair of access enable registers.

Each hardware resource partition supports a unique bus Master Tag Identifier (TAGID).

DMA_HRRz (z = 000-015)
DMA Channel Hardware Resource Register z

(1800_H + z * 0004_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
HRP	[1:0]	rw	**Hardware Resource Partition y** 00_B Hardware Resource Partition 0. 01_B Hardware Resource Partition 1. 10_B Hardware Resource Partition 2. 11_B Hardware Resource Partition 3.
RES	[31:2]	r	**Reserved** Read as 0; must be written with 0.

15.8.10 DMA Channel Suspend Registers

DMA Channel Suspend Enable Register z

The Suspend Enable Register enables/disables soft suspend mode capability.

DMA_SUSENRz (z = 000-015)
DMA Suspend Enable Register z

($1A00_H$ + z * 0004_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES															SUS EN
r															rw

Field	Bits	Type	Description
SUSEN	0	rw	**Channel Suspend Enable for DMA Channel z** This bit enables the channel suspend capability individually for each DMA channel z. 0_B DMA channel z is disabled for channel suspend mode. The DMA channel z does not react on an active suspend request signal SUSREQ 1_B DMA channel z is enabled for Soft-suspend Mode. If the suspend request signal SUSREQ becomes active, a DMA transaction of DMA channel z is stopped after the current DMA transfer has been finished Channel suspend mode can be terminated when SUSENz is written with 0.
RES	[31:1]	r	**Reserved** Read as 0; must be written with 0.

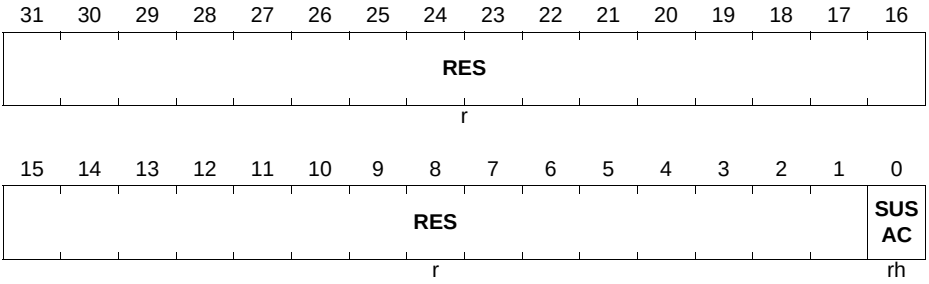
Note: Register is only reset by the Debug Reset.

DMA Suspend Acknowledge Register z

The Suspend Acknowledge Register indicates the DMA Channel soft suspend status.

DMA_SUSACRz (z = 000-015)
DMA Suspend Acknowledge Register z

($1C00_H$ + z * 0004_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
SUSAC	0	rh	**Channel Suspend Mode or Frozen State Active for DMA Channel z** This status bit indicates whether or not DMA channel z is in channel suspend mode or in the frozen state. 0_B DMA channel z is not in channel suspend mode, frozen state or internal actions are not yet finished after the channel suspend mode or frozen state was requested. 1_B DMA channel z is in channel suspend mode or frozen state.
RES	[31:1]	r	**Reserved** Read as 0; must be written with 0.

Note: This register is only reset by the Debug Reset.

15.8.11 DMA Transaction State Registers

DMA Channel Transaction State Register z

The Transaction State Register supports:

- The enabling and assertion of DMA channel hardware requests.
- The recording of a transaction request lost event for the channel.
- The setting, reporting and clearing of a DMA channel halt.
- The assertion of a DMA channel reset.

DMA_TSRz (z = 000-015)
DMA Transaction State Register z

($1E00_H$ + z * 0004_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES							HLT CLR	RES					CTL	DCH	ECH
r							w	r					w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES						HLT ACK	HLT REQ	RES				CH	TRL	HTRE	RST
r						rh	rwh	r				rh	rh	rh	rwh

Field	Bits	Type	Description
RST	0	rwh	**DMA Channel Reset** These bits force the DMA channel z to stop its current DMA transaction. Once set by software, this bit will be automatically cleared when the channel has been reset. Writing a 0 to RST has no effect. 0_B No action (write) or the requested channel reset has been reset (read). 1_B DMA channel z is stopped. More details see **Page 15-20**.

Field	Bits	Type	Description
HTRE	1	rh	**Hardware Transaction Request Enable State** 0_B Hardware transaction request for DMA Channel is disabled. An input DMA request will not trigger the channel. 1_B Hardware transaction request for DMA Channel is enabled. The transfers of a DMA transaction are controlled by the corresponding channel request line of the DMA requesting source. *Note: HTRE Operation* *4. Single Mode: HTRE is set to 0 when MExCHSR.TCOUNT is decremented and MExCHSR.TCOUNT = 0.* *5. HTRE can be enabled and disabled with TSRz.ECH and TSRz.DCH.* *6. HTRE is reset when a pattern match is detected.*
TRL	2	rh	**Transaction/Transfer Request Lost of DMA Channel** 0_B No request lost event has been detected for channel z. 1_B A new DMA request was detected while TSRz.CH=1 (request lost event). This bit is reset by software when writing a 1 to TSRz.CTL or by a channel reset (writing TSRz.RST = 1).
CH	3	rh	**Transaction Request State** 0_B No DMA request is pending for channel. 1_B A DMA request is pending for channel. CH is reset when a pattern match is detected.
HLTREQ	8	rwh	**Halt Request** The halt request is a status bit that remains active until it is cleared by TSRz.HALTCLR 0_B No action. 1_B Halt request.
HLTACK	9	rh	**Halt Acknowledge** Halt acknowledge status from DMA channel.
ECH	16	w	**Enable Hardware Transfer Request** see table below

Field	Bits	Type	Description
DCH	17	w	**Disable Hardware Transfer Request** see table below
CTL	18	w	**Clear Transaction Request Lost for DMA Channel z** Software clear of the DMA channel transaction request lost interrupt flag TSRz.TRL 0_B No action 1_B Clear DMA channel transaction request lost flag TSRz.TRL Reading this bit returns a 0.
HLTCLR	24	w	**Clear Halt Request and Acknowledge** Software write only active high clear of halt request TSRz.HALTREQ and halt acknowledge status bit TSRz.HALTACK 0_B No action. 1_B Halt clear. Reading this bit returns a 0.
RES	[7:4], [15:10], [23:19], [31:25]	r	**Reserved** Read as 0; must be written with 0.

Note: The DMA channel transaction/transfer request lost of a DMA channel is permanently available state that detects the loss of a DMA service request. If a transaction/transfer request lost is detected then an error interrupt trigger will be generated when the DMA channel z becomes active in a DMA sub-block x and if DMARAM bit ADICRz.ETRL enable bit is high.

DMA Channel Hardware Transaction/Transfer Request Conditions

Table 15-19 Conditions to Set/Reset the Bits TSRz.HTRE

TSRz.ECH	TSRz.DCH	Transaction Finishes[1)] for Channel z	Modification of TSRz.HTRE
0	0	0	unchanged
1	0	0	set
X	1	X	reset
X	X	1	reset

1) In Single Mode only. In Continuous Mode, the end of a transaction has no impact.

15.8.12 DMA Transaction Control Set

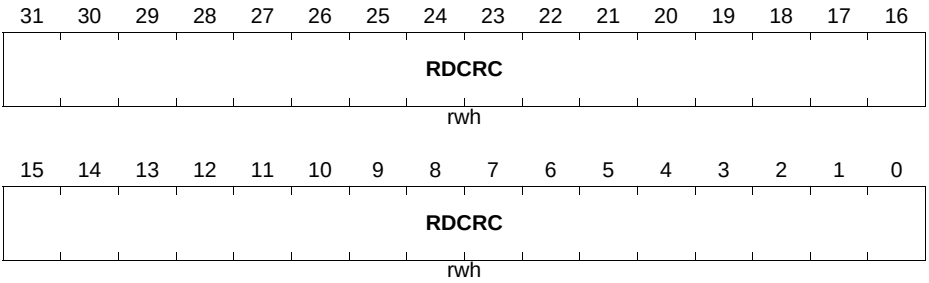
The composition of a DMA transaction for each DMA channel is defined the transaction control set stored in the DMARAM. Each transaction control set consists of 8 X 32-bit words:

- DMA_RDCRCRz, DMA_SDCRCRz[1)], DMA_SADRz, DMA_DADRz and DMA_SHADRz store 32-bit CRC or address values.
- DMA_ADICRz and DMA_CHCFGRz store DMA channel configuration control bits.
- DMA_CHCSRz stores active channel status and write only active triggers.

DMA Channel Read Data CRC Register

DMA_RDCRCRz (z = 000-015)
DMA Channel Read Data CRC Register z

(2000_H + z * 20_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
RDCRC	[31:0]	rwh	**Read Data CRC** This bit field stores the CRC32 ethernet polynomial checksum for read data of an inactive DMA channel.

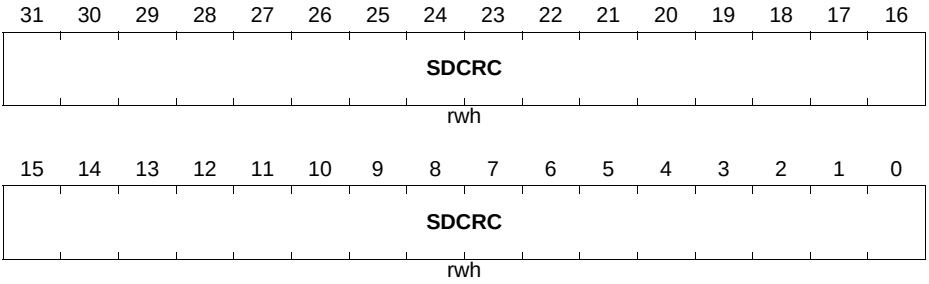
1) The Source and Destination Address CRC can detect failures in the DMA fetch configuration information and updates to the dynamic channel information.

DMA Channel Source and Destination Address CRC Register

DMA_SDCRCRz (z = 000-015)

DMA Channel Source and Destination Address CRC Register z

(2004_H + z * 20_H) **Reset Value: 0000 0000_H**

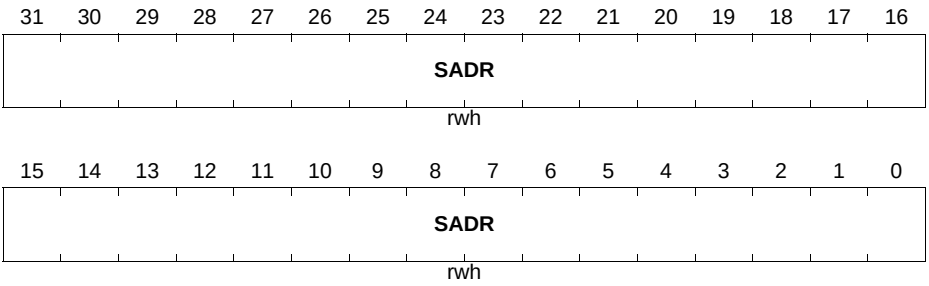


Field	Bits	Type	Description
SDCRC	[31:0]	rwh	**Source and Destination Address CRC** This bit field stores the CRC32 ethernet polynomial checksum for the source and destination address of an inactive DMA channel.

DMA Channel Source Address Register

The Source Address Register stores the source address of an inactive DMA channel.

DMA_SADRz (z = 000-015)
DMA Channel Source Address Register z
(2008_H + z * 20_H) **Reset Value: 0000 0000_H**



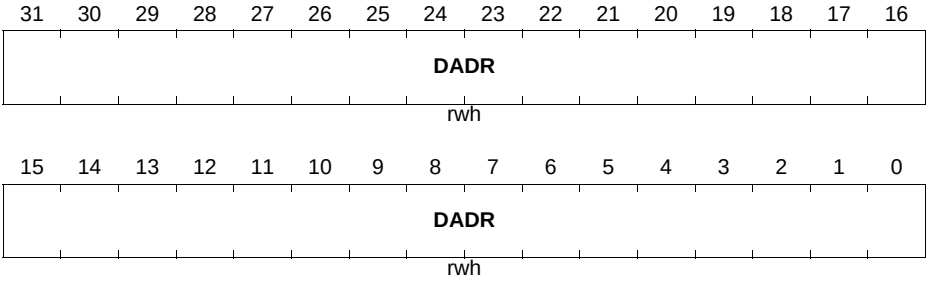
Field	Bits	Type	Description
SADR	[31:0]	rwh	**Source Address** This bit field holds the actual 32-bit source address of an inactive DMA channel.

Note: If a DMA channel is configured to execute 16-bit DMA moves then the source address register should be aligned to a 16-bit start source address.

DMA Channel Destination Address Register

The Destination Address Register stores the destination address of an inactive DMA channel.

DMA_DADRz (z = 000-015)
DMA Channel Destination Address Register x
($200C_H$ + z * 20_H) **Reset Value: 0000 0000_H**



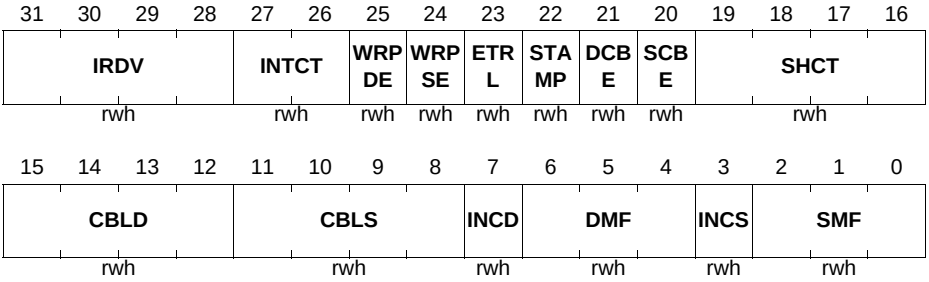
Field	Bits	Type	Description
DADR	[31:0]	rwh	**Destination Address** This bit field holds the actual 32-bit destination address of an inactive DMA channel.

Note: If a DMA channel is configured to execute 16-bit DMA moves then the destination address register should be aligned to a 16-bit start destination address.

DMA Channel Address and Interrupt Control Register

The Address and Interrupt Control Register stores bits to update source and destination addresses and the generation of DMA channel traffic management interrupt triggers.

DMA_ADICRz (z = 000-015)
DMA Channel Address and Interrupt Control Register x
(2010_H + z * 20_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
SMF	[2:0]	rwh	**Source Address Modification Factor** This bit field stores the Source Address Modification Factor of an inactive DMA channel.
INCS	3	rwh	**Increment of Source Address** This inactive DMA channel control bit determines if the source address is decremented or incremented.
DMF	[6:4]	rwh	**Destination Address Modification Factor** This bit field stores the Destination Address Modification Factor of an inactive DMA channel.
INCD	7	rwh	**Increment of Destination Address** This bit field stores the Destination Address Modification Factor for an inactive DMA channel.
CBLS	[11:8]	rwh	**Circular Buffer Length Source** This bit field stores circular buffer address update control of an inactive DMA channel.
CBLD	[15:12]	rwh	**Circular Buffer Length Destination** This bit field stores circular buffer address update control of an inactive DMA channel.
SHCT	[19:16]	rwh	**Shadow Control** Inactive DMA channel shadow control.

Field	Bits	Type	Description
SCBE	20	rwh	**Source Circular Buffer Enable** Inactive DMA channel source circular buffer enable.
DCBE	21	rwh	**Destination Circular Buffer Enable** Inactive DMA channel destination circular buffer enable.
STAMP	22	rwh	**Time Stamp** Enable appendage of time stamp.
ETRL	23	rwh	**Enable Transaction Request Lost Interrupt** Inactive DMA channel bit to enable the generation of an error interrupt service request.
WRPSE	24	rwh	**Wrap Source Enable** Inactive DMA channel source buffer interrupt trigger enable/disable.
WRPDE	25	rwh	**Wrap Destination Enable** Inactive DMA channel destination buffer interrupt trigger enable/disable.
INTCT	[27:26]	rwh	**Interrupt Control** Inactive DMA channel interrupt control.
IRDV	[31:28]	rwh	**Interrupt Raise Detect Value** Inactive DMA channel interrupt threshold value.

Shadow Address Register Read Only Mode

If ADICRz.SHCT = 0001_B or ADICRz.SHCT = 0010_B then source or destination address buffering is selected and SHADRz is written when a transaction is running. The SHADRz is automatically set to 0000 0000_H when the shadow transfer takes place. The user can read the shadow register in order to detect if the shadow transfer has already taken place. If the value in SHADRz is 0000 0000_H, no shadow transfer can take place and the corresponding address register is modified according to the circular buffer rules.

Shadow Address Register Write Enable Mode

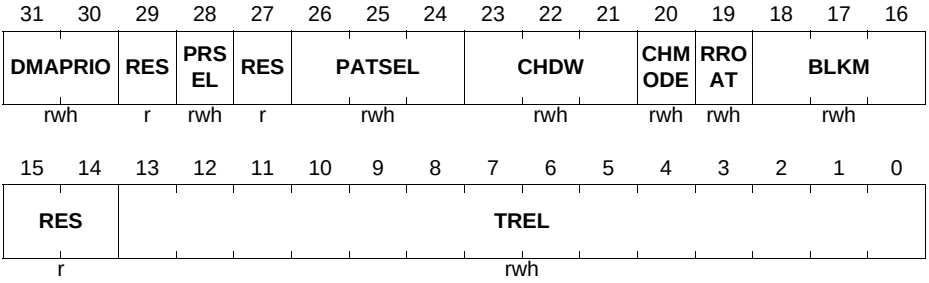
If ADICRz.SHCT = 0101_B or ADICRz.SHCT = 0110_B then the shadow register SHADRz can be directly written. The value stored in the MExSHADR is not modified when the shadow transfer takes place, the shadow mechanism remains active and the shadow transfer will be repeated until Channel z is reset or until the value in SHADRz is 0000 0000_H, is written into the shadow register.

DMA Channel Configuration Register

The Channel Configuration Register stores control and configuration data.

DMA_CHCFGRz (z = 000-015)
DMA Channel Configuration Register z

(2014_H + z * 20_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
TREL	[13:0]	rwh	**Transfer Reload Value** Inactive DMA channel transfer reload value.
BLKM	[18:16]	rwh	**Block Mode** Inactive DMA channel block mode control bit field.
RROAT	19	rwh	**Reset Request Only After Transaction** Inactive DMA channel control bit to reset the request state flag.
CHMODE	20	rwh	**Channel Operation Mode** Inactive DMA channel control bit to reset condition of the hardware transaction request enable bit.
CHDW	[23:21]	rwh	**Channel Data Width** Inactive DMA channel data width bit field.
PATSEL	[26:24]	rwh	**Pattern Select** Inactive DMA channel pattern select bit field.
PRSEL	28	rwh	**Peripheral Request Select** Inactive DMA channel hardware request trigger control.
DMAPRIO	[31:30]	rwh	**DMA Priority** Inactive DMA channel priority bit field.

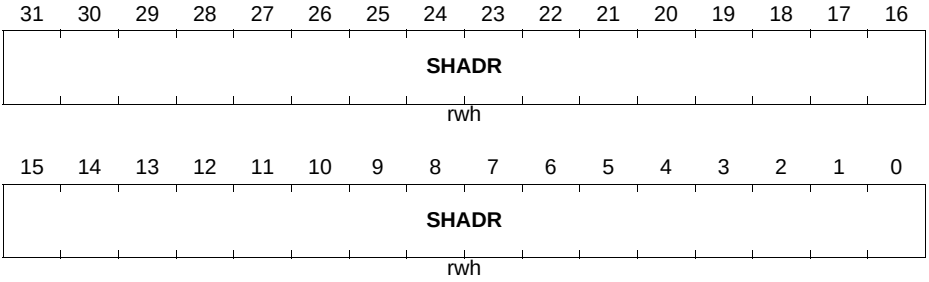
Field	Bits	Type	Description
RES	[15:14], 27, 29	r	**Reserved** Read as 0; must be written with 0.

DMA Channel Shadow Address Register

The Shadow Address Register stores the 32-bit shadow address of an inactive DMA channel.

DMA_SHADRz (z = 000-015)
DMA Channel Shadow Address Register z

(2018_H + z * 20_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
SHADR	[31:0]	rwh	**Shadowed Address** This bit field stores the 32-bit shadow address of an inactive DMA channel.

DMA Channel Control and Status Register

The Channel Control and Status Register contains the current transfer count, channel reset, pattern detection compare result and the status of traffic management interrupt triggers.

DMA_CHCSRz (z = 000-015)
DMARAM Channel Control and Status Register z
($201C_H$ + z * 20_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCH	RES			SIT	CICH	CWRP	SWB	FROZEN	BUFFER	RES		IPM	ICH	WRPD	WRPS
w	r			w	w	w	w	rwh	rh	r		rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LXO	RES	TCOUNT													
rh	r	rh													

Field	Bits	Type	Description
TCOUNT	[13:0]	rh	**Transfer Count Status** DMA transfer count for an inactive DMA channel.
LXO	15	rh	**Old Value of Pattern Detection** Inactive DMA channel compare result.
WRPS	16	rh	**Wrap Source Buffer** Inactive DMA channel wrap-around of source buffer interrupt trigger flag.
WRPD	17	rh	**Wrap Destination Buffer** Inactive DMA channel wrap-around of destination buffer interrupt trigger flag.
ICH	18	rh	**Interrupt from Channel** Inactive DMA channel interrupt trigger flag.
IPM	19	rh	**Pattern Detection from Channel** Inactive DMA channel pattern detection flag.
BUFFER	22	rh	**DMA Double Buffering Active Buffer** This bit is active during DMA double buffering and indicates which buffer is read or filled. 0_B Buffer 0 read or filled by DMA. 1_B Buffer 1 read or filled by DMA.

Field	Bits	Type	Description
FROZEN	23	rwh	**DMA Double Buffering Frozen Buffer** This bit is active during DMA double buffering and indicates that one of the double buffers is frozen and available for a cyclic software task. 0_B Buffer is not frozen. 1_B Buffer is frozen. *Note: FROZEN bit must only be cleared by software.*
SWB	24	w	**DMA Double Buffering Switch Buffer** When double buffering is selected by programming ADICRz.SHCT then the control bit is used to re-direct data from one buffer to the other buffer. 0_B No action. 1_B Switch from buffer. *Note: If a Linked List mode (ADICRz.SHCT[3:2] = 11) is configured then SWB must be 0.*
CWRP	25	w	**Clear Wrap Buffer Interrupt z** Software clear of both the DMA channel source and destination wrap buffer interrupts stored in DMARAM at locations CHSRz.WRPS and CHSRz.WRPD 0_B No action. 1_B Bits CHSRz.WRPS and CHSRz.WRPD are reset. *Note: If DMA channel z is active in Move Engine x then clear bit fields MExCHSR.WRPS and MExCHSR.WRPD* Reading this bit returns a 0.
CICH	26	w	**Clear Interrupt for DMA Channel z** Software clear of the DMA channel interrupt flags stored in DMARAM locations CHSRz.ICH and CHSRz.IPM 0_B No action. 1_B Bits MExCHSR.ICH and MExCHSR.IPM are reset. *Note: If DMA channel z is active in Move Engine x then clear bit fields MExCHSR.ICH and MExCHSR.IPM* Reading this bit returns a 0.

Field	Bits	Type	Description
SIT	27	w	**Set Interrupt Trigger for DMA Channel z** 0_B No action. 1_B Channel z interrupt trigger will be activated. Reading this bit returns a 0. *Note: If a Linked List mode (ADICRz.SHCT[3:2] = 11) is configured then SIT must be 0.*
SCH	31	w	**Set Transaction Request for DMA Channel** 0_B No action. 1_B A transaction for DMA channel z is requested. When setting SCH, TSRz.CH becomes set to indicate that a DMA request is pending for DMA channel z.
RES	14, [21:20], [30:28]	r	**Reserved** Read as 0; must be written with 0.

15.9 Use Cases

This section provides a code example for the DMA module to give an overview about the core functionality. The example realizes a one word (32 bit-length) single data transfer from a data source location to a data destination location without intervention of the CPU. The transfer gets triggered by software in this example. The module also supports hardware triggering, please see **Chapter 15.4.3.3** for more details.

There will be no interrupt or further functions created in this example, please see the respective registers for more details on that. The examples uses DMA channel 000 (z=000).

Step description to initialize and trigger a data transfer via DMA:

(Line 1) The 32-bit data source address (DMA_SADR000_ADD) gets stored in the source address register SADR000 (see register definition **DMA_SADR**).

(Line 2) The 32-bit data destination address (DMA_DADR000_ADD) gets stored in the destination address register DADR000 (see register definition **DMA_DADR**).

(Line 3) The channel data width is defined in the channel configuration register CHCR000 (see register definition **DMA_CHCFGR**). Setting DMA_CHCFGR000.[23:21] = 010_B sets the channel data width to 32 bit.

(Line 4) DMA hardware transfer requests are disabled in the transaction state register TSR000 (see register definition **DMA_TSR**). Writing DMA_TSR000.[17] = 1 disables the hardware transfer requests.

(Line 5) This line starts the data transfer between the source and destination memory address.

Note: the declaration of DMA_SADR000_ADD and DMA_DADR000_ADD must be done by the user.

Initialization Code for 1 ms timer interrupt

```
(1)  DMA_SADR000 = DMA_SADR000_ADD; //data source add
(2)  DMA_DADR000 = DMA_DADR000_ADD; //data destination add
(3)  DMA_CHCFGR000.U |= (010 << 21);
(4)  DMA_TSR000.U |= (1 << 17); //disable hardware requests
(5)  DMA_CHCSR000.U |= (1 << 31); //start transfer
```

Note: The transfer can be started everywhere in the program.

16 Interrupt Router (IR)

The chapter describes the Interrupt Router (IR) module that schedules on TC21x/TC22x/TC23x Interrupts (here called service requests) from external resources, internal resources and SW to the CPU and the DMA module (here called service provider).

Topics covered include the architecture of the interrupt system, interrupt system configuration, and the interrupt operations of the TC21x/TC22x/TC23x peripherals.

16.1 Overview

An interrupt request can be serviced either by the CPUs or by the DMA module. Interrupt requests are called “service requests” rather than “interrupt requests” in this document because they can be serviced by either one of the service providers.

The interrupt system in the TC21x/TC22x/TC23x is realized in the Interrupt Router module which includes the Service Request Nodes (SRNs), the Interrupt Control Units (ICUs) and additional functionality for SW development support.

As shown in **Figure 16-1**, each TC21x/TC22x/TC23x module that can generate service requests is connected to one or more Service Request Nodes (SRNs) in the central Interrupt Router module. The Interrupt Router module includes also several general purpose Service Request Nodes (SRNs) that can be used for software (SW) triggered service requests.

Each SRN contains a Service Request Control Register (SRC) to configure the service request regarding e.g. priority, mapping to one of the available service providers.

Each SRN is connected to all ICUs in the interrupt router module where the SRNs control register setting defines the target service provider and the priority of the service request.

Each ICU handles the interrupt arbitration among competing service requests from SRNs that are mapped to the ICU.

Each ICU is connected to one service provider (CPU or DMA module) where the ICU offers the valid winning service request/SRN of an arbitration round and the service provider signals back to the ICU when and which service request it is processing.

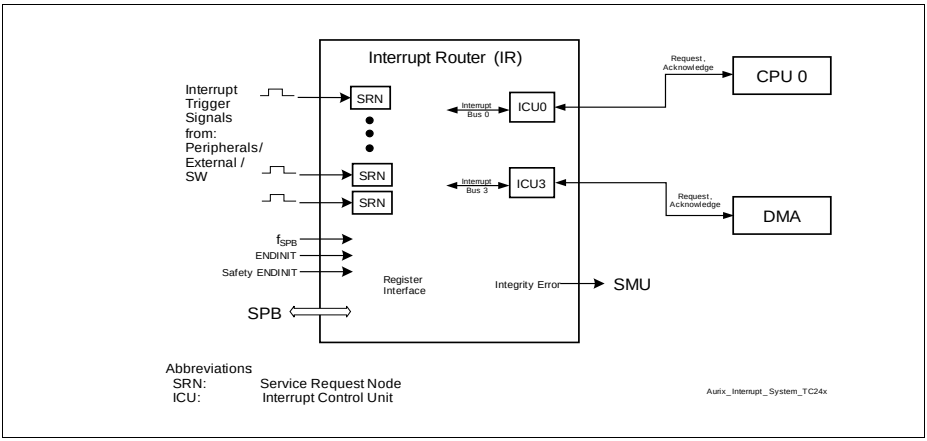


Figure 16-1 Block Diagram of the TC21x/TC22x/TC23x Interrupt System

16.2 Features

- Interrupt System with support of up to 512 / 1024 service requests
- Support of up to 255 service request priority levels per ICU[1)] / Service Provider
- Support of up to 16 ICU / service provider
- Each CPU and the DMA module with dedicated ICU
- Low latency arbitration - three / four clocks[2)] from receipt of an service request to sending it to the service provider
- Each Service Request with a dedicated Service Request Node (SRN)
- Each SRN with a programmable 8-bit priority vector[1)]
- Each SRN can be mapped to one of the implemented ICUs / Service Providers
- SRNs are cleared automatically by hardware on interrupt acknowledge by the configured service provider
- Interrupt System Integrity support
- Four General Purpose Service Requests (GPSR) per CPU that can be used as Software Interrupts (not assigned to peripherals or external interrupts)
- Mechanism to signal General Purpose Service Requests (Software Interrupts) simultaneously to multiple Service Providers (Service Request Broadcast Registers, SRB)

1) Max. 255 of the implemented service requests can be mapped to one CPU

2) Depends on the complexity and the clock frequency of the Interrupt Router. Details for the implementation are described in the chapter Module Implementation

- Priority dependent masking of service requests (for CPUs, related control registers included in the CPUs)
- External Interrupts with filter modes and trigger modes (to e.g. falling edge, rising edge, high or low level). Modes can be configured during runtime[1)]

16.3 Service Request Nodes (SRN)

Each Service Request node (SRN) inside the Interrupt Router module contains a Service Request Control (SRC) Register and interface logic that connects it to the triggering unit outside the Interrupt Router module and to the interrupt arbitration buses inside the Interrupt Router (see also: **Figure 16-1**).

16.3.1 Service Request Control Registers

All Service Request Control Registers in the Interrupt Router module have the same format. In general, these registers contain:

- Enable/disable information
- Service Request Set bit and Service Request Clear bit
- Service Request Priority Level vector (8-bit)
- Service provider destination
- Service request status bit
- Software-initiated service request set and reset bits
- Integrity errors signalled to the Safety Monitor Unit (SMU)
- Interrupt Sticky and Overflow bits

Besides being activated by the associated triggering unit through hardware, each SRN can also be set or reset by software via two software-initiated service request control bits.

The description given in this chapter characterizes all Service Request Control Registers of the TC21x/TC22x/TC23x. Information on peripheral module interrupt functions such as enable or request flags is provided in the corresponding sections of the module chapters.

16.3.1.1 General Service Request Control Register Format

Service Request Control Register (SRC)

The description given in this chapter characterizes all Service Request Control Registers of the TC21x/TC22x/TC23x. Information on peripheral module interrupt functions such as enable or request flags is provided in the corresponding sections of the module chapters.

1) External Interrupt logic and related control registers are described in the System Control Unit (SCU) chapter, External Request Unit (ERU)

SRC

Service Request Control Register (xx_H) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**SWS CLR**	**SWS**	**IOVC LR**	**IOV**	**SET R**	**CLR R**	**SRR**	**0**			**ECC**				
r	w	rh	w	rh	w	w	rh	r			rwh				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				**TOS**	**SRE**	**0**		**SRPN.[IR_Nr_Of_Interrupts_003]**							
r				rw	rw	r		rw							

Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number** 00_H Service request is on lowest priority 01_H Service request is one before lowest priority … … FF_H Service request is on highest priority *Note: For a CPU 01H is the lowest priority as 00H is never serviced. For the DMA 00H triggers channel 0* *Note: For DMA, SRPN must be =< the highest implemented DMA channel number*
SRE	10	rw	**Service Request Enable** 0_B Service request is disabled 1_B Service request is enabled
TOS	11	rw	**Type of Service Control** 0_H CPU0 service is initiated 1_H DMA service is initiated

Field	Bits	Type	Description
ECC	[20:16]	rwh	**ECC** The SRC.ECC bit field will be updated by the SRN under the following conditions: • write or Read-Modify-Write to SRC[31:0] • write to SRC[15:0] (16-bit write) • write to SRC[15:8] or write to SRC[7:0] (byte write) In case of a 32 bit write to SRC, the ECC bit field will be updated with the calculated ECC, the data written to ECC bit field will be ignored. ECC encoding covers new SRPN, TOS, SRE values and the internal index number of the written SRN. There is no permanent ECC check. ECC check will be done whenever the SRN with a pending interrupt was accepted by the selected (TOS) service provider as next service request to be processed. For a check of the error detection mechanisms ECC errors can be inserted (ECC bit field modified) by: writing directly to the ECC bit field. • writing to SRC[23:16] (8-bit write) • writing to SRC[31:16] (16-bit write) *Note: In the current implementation the ECC code is only used for error detection. Detected errors are reported to the SMU but not corrected.*
SRR	24	rh	**Service Request Flag** 0_B No service request is pending 1_B A service request is pending
CLRR	25	w	**Request Clear Bit** CLRR is required to reset SRR. 0_B No action 1_B Clear SRR; bit value is not stored; read always returns 0; no action if SETR is set in parallel.
SETR	26	w	**Request Set Bit** SETR is required to set SRR. 0_B No action 1_B Set SRR and SWS; SRR bit value is not stored; read always returns 0; no action if CLRR is set in parallel.

Field	Bits	Type	Description
IOV	27	rh	**Interrupt Trigger Overflow Bit** The IOV bit is set by HW if a new service request was triggered via interrupt trigger or SETR bit while the SRN has still an pending service request 0_B No Interrupt Trigger Overflow detected 1_B Interrupt Overflow Detected.
IOVCLR	28	w	**Interrupt Trigger Overflow Clear Bit** IOVCLR is required to reset IOV. 0_B No action 1_B Clear IOV; bit value is not stored; read always returns 0.
SWS	29	rh	**SW Sticky Bit** The Software Sticky Bit is set when the SRR bit has been set via the SETR (Request Set Bit). 0_B No interrupt was initiated via SETR 1_B Interrupt was initiated via SETR This bit can be cleared by writing with 1. Writing with 0 has no effect.
SWSCLR	30	w	**SW Sticky Clear Bit** SWSCLR is required to reset SWS. 0_B No action 1_B Clear SWS; bit value is not stored; read always returns 0.
0	31, [23:21], [15:12], [9:8]	r	**Reserved** Read as 0; should be written with 0.

16.3.1.2 Changing the SRN configuration

All Service Request Nodes are disabled per default. To use a Service Request Node, it has to be configured and enabled by setting the SRC.SRE bit=´1´.

The Service Request Nodes can be configured regarding the interrupt service provider target (SRC.TOS) and regarding the service request priority number (SRC.SRPN).

Once an SRN is enabled, the TOS and/or SRPN bit fields can be changed by using the following sequence:

- disable the SRN (SRC.SRE=´0´) (write to SRC.SRE)
- check that the SRN is disabled (read back SRC.SRE and check for SRE=´0)
- check the register LWSRx (read/poll LWSRx, see note below):
- if LWSRx.STAT=´1´ and LWSRx.SRPN and LWSRx.ECC are equal to the (old) SRC value check again
- if LWSRx.STAT=´0´ or LWSRx.SRPN or LWSRx.ECC are not equal to the old SRC values anymore go on with the next step (change SRC value)
- change the SRC.TOS and/or SRC.SRPN bit field
- enable the SRN (SRC.SRE=´1´) (write to SRC.SRE)

The read/poll of LWSRx is mandatory before changing SRC.TOS and/or SRC.SRPN. Otherwise the last interrupt before the SRN disabling could still be taken (enter) for execution in the CPU. It could be acknowledged later by the CPU with the side effect that the first interrupt of the just re-configured SRN can get lost. The time between enter and acknowledge is not deterministic if the ISP is a TriCore. If the ISP is the DMA module, enter and acknowledge are signalled in the same cycle. The ´x´ in LWSRx means that the LWSR register related to the TOS setting has to be checked (TOS=0 -> LWSR0, TOS=1 -> LWSR1, ...).

16.3.1.3 Protection of the SRC Registers

The SRC register is write protected via an On Chip Bus Master TAG-ID protection (see **Chapter 16.6.1**). This protection is controlled via the Interrupt Router control registers ACCEN10 and ACCEN00.

- SRC[31:16] is write protected by ACCEN00
- SRC[15:0] is write protected by ACCEN10

This implementation allows to define a first group of TAG ID(s) that are allowed to modify the configuration of the Service Request Nodes (TOS, SRPN, SRE), and a second group of TAG IDs that are allowed to modify the Service Request Nodes control bits (set/clear of SW Interrupts, clear of Sticky and Overflow bits)

Write access to SRC[31:16] with access protection violation

When an SRC register is written with a 32 bit data access, only the SRC register parts without ACCEN protection violation will be updated:

- Violation for SRC[31:16] (ACCEN00) and SRC[15:0] (ACCEN10)
- no update of the SRC register, Alarm send to SMU
- Violation for SRC[31:16] (ACCEN00)
- no update of SRC[31:16], SRC[15:0] updated, Alarm send to SMU
- Violation for SRC[31:16] (ACCEN10)
- SRC[31:16] updated, no updated of SRC[15:0], Alarm send to SMU

16.3.1.4 Request Set and Clear Bits (SETR, CLRR)

The SETR and CLRR bits allow software to set or clear the service request bit SRR.

- Writing a 1 to SETR causes bit SRR to be set to 1
- Writing a 1 to CLRR causes bit SRR to be cleared to 0
- Writing a 1 to SETR and CLRR at the same time, SRR is not changed
- The value written to SETR or CLRR is not stored
- Writing a 0 to these bits has no effect
- These bits always return 0 when read

16.3.1.5 Enable Bit (SRE)

The SRE bit enables an interrupt to take part in the arbitration for the selected service provider. It does not enable or disable the setting of the request flag SRR; the request flag can be set by hardware or by software (via SETR) independent of the state of the SRE bit. This allows service requests to be handled automatically by hardware or through software polling.

If SRE = 1, pending service requests are passed on to the designated service provider for interrupt arbitration. The SRR bit is automatically set to 0 by hardware when the service request is acknowledged by the Interrupt Service Provider. It is recommended that in this case, software should not modify the SRR bit to avoid unexpected behavior due to the hardware controlling this bit.

If SRE = 0, pending service requests are not passed on to service providers. Software can poll the SRR bit to check whether a service request is pending. To acknowledge the service request, the SRR bit must then be reset by software by writing a 1 to CLRR.

Note: In this document, ‘active source’ means an SRN whose Service Request Control Register has its request enable bit SRE set to 1 to allow its service requests to participate in interrupt arbitration.

16.3.1.6 Service Request Flag (SRR)

When set, the SRR flag indicates that a service request is pending. It can be set or reset directly by hardware or indirectly through software using the SETR and CLRR bits. Writing directly to this bit via software has no effect.

SRR can be set or cleared either by hardware or by software regardless of the state of the enable bit SRE. However, the request is only forwarded for service if the enable bit is set. If SRE = 1, a pending service request takes part in the interrupt arbitration of the service provider selected by the device's TOS bit field. If SRE = 0, a pending service request is excluded from interrupt arbitrations.

SRR is automatically reset by hardware when the service request is acknowledged and serviced. Software can poll SRR to check for a pending service request. SRR must be reset by software in this case by writing a 1 to CLRR.

Note: Clearing a pending service request flag SRR and enabling the corresponding service request node (SRN) should be done in two steps / two writes: first clear the SRR flag (SRC.CLRR), than enable the (SRC.SRE).

16.3.1.7 Type-Of-Service Control (TOS)

There are two service providers for service requests in the TC21x/TC22x/TC23x, the CPU and the DMA module. The TOS bit is used to select to which of the available interrupt service provider an service request has to be forwarded.

Note: Before modifying an TOS or an SRPN bit field, the corresponding SRN must be disabled (SRE = 0).

16.3.1.8 Service Request Priority Number (SRPN)

The 8-bit Service Request Priority Number (SRPN) indicates the priority of a service request with respect to other sources requesting service from the same service provider, and with respect to the priority of the service provider itself.

SRPN Rules:

- Each active source selecting the same service provider must have a unique SRPN value to differentiate its priority
- Active sources selecting different service provider could use same SRPN values
- If a source is not active – meaning its SRE bit is 0 – no restrictions are applied to the service request priority number
- The SRPN is used by service providers to select an Interrupt Service Routine (ISR)) or DMA channel to service the request

Service Provider is a CPU:

ISRs are associated with Service Request Priority Numbers by an Interrupt Vector Table located in each CPU. This means that the CPU Interrupt Vector Table is ordered by priority number. This is unlike traditional interrupt CPU architectures in which their interrupt vector tables are ordered by the source of the interrupt. The CPU Interrupt Vector Tables allow a single peripheral to have multiple priorities for different purposes.

Note: For a CPU, the SRPN value of 0000_H is a special value and must not be used for service requests mapped to a CPU.

Note: TC1.6E and TC1.6P CPUs are providing a flexible interrupt table alignment with a configurable vector spacing of 8 byte or 32 byte. Pls. see also CPU chapter.

Service Provider is the DMA:

The complete SRPN number is used for the arbitration of service requests the DMA module. Within the module, the DMA channels are associated with the Service Request Priority Number by the SRPN LSBs of the SRPN.

Only the SRPN numbers 0 ... max_channel_number will result in a trigger of the related DMA channel. SRPN numbers > max_channel number do not result in a DMA channel trigger nor any other signalling.

Examples:

- In case of an 16 channel DMA module the SRPN number 00H will trigger the channel 0, 07H will trigger the channel 7. All SRPN > 0FH will not result in a channel trigger.
- In case of an 64 channel DMA module the SRPN number 00H will trigger the channel 0, 17H will trigger the channel 23 an 3FH. will trigger channel 64. All SRPN > 3FH will not result in a channel trigger.

16.3.1.9 ECC Encoding (ECC)

The SRC.ECC bit field will be updated by the SRN under the following conditions:

- write or Read-Modify-Write to SRC[31:0]
- write to SRC[15:0] (16-bit write)
- write to SRC[15:8] or write to SRC[7:0] (byte write)

In case of a 32-bit write or a Read-Modify-Write to the SRC, the ECC bit field will be updated with the calculated ECC, the data written to ECC bit field will be ignored.

ECC encoding covers the new values of SRC.SRPN, SRC.TOS, SRC.SRE and the internal 10 bit index number of the written SRN.

There is no permanent ECC check. ECC check will be checked whenever the SRN with an pending service request was accepted by the selected (TOS) service provider as next service request to be processed.

For a check of the error detection mechanisms ECC errors can be inserted (ECC bit field modified) by:

- writing to SRC[23:16] (byte write)
- writing to SRC[31:16] (16-bit write)

Note: In the current implementation the ECC code is only used for error detection. Detected errors are reported to the SMU but not corrected.

ECC Code

The ECC code used for the IR Error Detection mechanism is a Hsiao 22_5 code with DED (double error detection) capability:

```
GEN_ENC22_5 : if (word_width_g/(nb_mems_g*ecc_granularity_g)) =
22 and (nb_check_bits_g = 5) generate
    cmr22_5: for i in 0 to nb_check_bits_g - 1 generate
      CODE_MATRIX_ROWS(i)     <= code_rows_22_5(i);
    end generate;
  end generate;


type rows_22_5_t is array (4 downto 0) of std_ulogic_vector(21
downto 0);
  constant code_rows_22_5 : rows_22_5_t :=
("0001001011001011011011",
"0010010101010101101101",
"0100100110100110110110",
"1000111000111000111111",
"1111000000111111000111");
```

16.3.1.10 Interrupt Trigger Overflow Bit (IOV)

The IOV bit is set by HW if both conditions are true:

- service request is pending
- a new service request is triggered via interrupt trigger or SETR bit

16.3.1.11 Interrupt Trigger Overflow Clear Bit (IOVCLR)

The Interrupt Trigger Overflow Clear Bit can be used to clear the IOV bit. The IOV bit is cleared by writing a ´1´ to the IOVCLR bit.

16.3.1.12 SW Sticky Bit (SWS)

The Software Sticky Bit is set when the SETR (Request Set Bit) is written with 1.

16.3.1.13 SW Sticky Clear Bit (SWSCLR)

The Software Sticky Clear Bit can be used to clear the SWS bit. The SWS bit is cleared by writing a ´1´ to the SWSCLR bit.

16.4 Interrupt Control Unit (ICU)

The Interrupt Router module includes one ICU per service provider (CPUs and DMA module) where each ICU is related to one service provider. SRNs can be mapped to one of the ICUs via the SRNs SRCx.TOS register bit field (see also: **Figure 16-1**).

The Interrupt Control Units (ICU):

- Manages the arbitration among competing service requests from SRNs that are mapped to the ICU
- Provides the winner of the arbitration round to the service provider
- Receives the information from the service provider which service request was accepted
- Checks the accepted service request information (ECC check)
- Signals integrity errors to the Safety Monitor Unit (SMU)
- Manages the clearing of acknowledged service requests in the related SRNs

Note: In the current implementation the ECC code is only used for error detection. Detected errors are reported to the SMU but not corrected.

16.4.1 ICU Control Registers

This section describes the Interrupt Control Unit (ICU) registers. Each ICU includes two control registers:

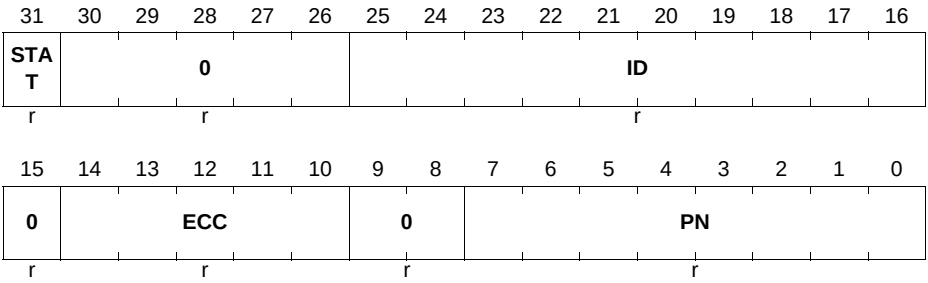
- Latest Winning Service Request register (LWSR) provides information about the winner of the last service request arbitration round
- Last Acknowledged Service Request register (LASR) provides information about the last service request that was accepted by the service provider

16.4.1.1 Latest Winning Service Request Register (LWSR)

Latest Winning Service Request (LWSR)

The Latest Winning Service Request register provides information about the winner of the last arbitration round. The register bit fields are representing what is provided by the ICU to the Interrupt Service Provider.

LWSR
Latest Winning Service Request (xx_H) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
PN	[7:0]	r	**Latest Winner Priority Number** This bit field shows the Priority Number of the service request that won the last arbitration round. This bit field is only valid if STAT is set to 1
ECC	[14:10]	r	**Latest Winner ECC** This bit field shows the ECC field (SRN.ECC) that was transferred from the last winning SRN to the ICU. This bit field is only valid if STAT is set to 1. *Note: In the current implementation the ECC code is only used for error detection. Detected errors are reported to the SMU but not corrected.*
ID	[25:16]	r	**Latest Winner ID** This bit field shows the ID number of the last winning SRN. This bit field is only valid if STAT is set to 1

Field	Bits	Type	Description
STAT	31	r	**LWSR Register Status** The STAT register indicates if the PN, ECC and ID bit fields are still valid. They are still valid if the interrupt from the SRN identified by ID is still pending. If the ICU does not have an winner because no interrupt is pending or not yet arbitrated than it clears the STAT bit. 0_B LWSR bit fields are not valid 1_B LWSR bit fields are valid
0	[30:26], 15, [9:8]	r	**Reserved** Read as 0; should be written with 0.

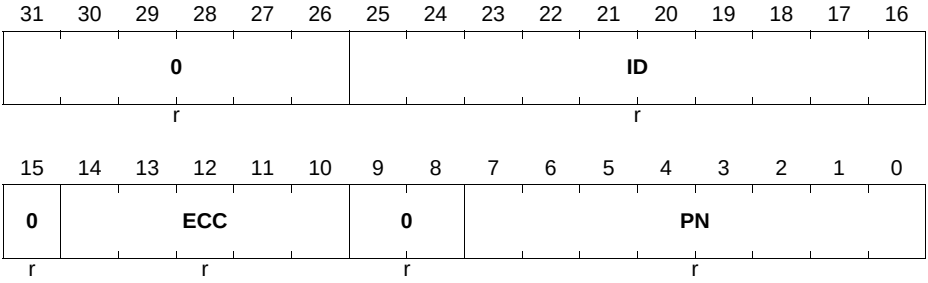
16.4.1.2 Last Acknowledged Service Request Register (LASR)

Last Acknowledged Service Request (LASR)

The Last Acknowledged Service Request register is representing the information that was sent by the Interrupt Service Provider together with the last acknowledge.

LASR

Last Acknowledged Service Request (xx_H) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
PN	[7:0]	r	**Last Acknowledged Service Request Priority Number** This bit field shows the Priority Number of the last acknowledged service request
ECC	[14:10]	r	**Last Acknowledged Interrupt ECC** This bit field shows the ECC value of the last acknowledged service request, as send by the service provider with acknowledge *Note: In the current implementation the ECC code is only used for error detection. Detected errors are reported to the SMU but not corrected.*
ID	[25:16]	r	**Last Acknowledged Interrupt SRN ID** This bit field shows the ID number of the last acknowledged service request,, as send by the service provider with acknowledge
0	[31:26], 15, [9:8]	r	**Reserved** Read as 0; should be written with 0.

16.4.1.3 Error Capture Register (ECR)

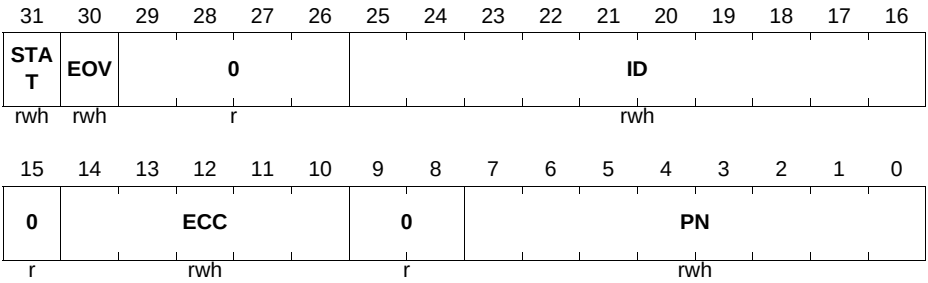
Error Capture Register (ECR)

The Error Capture Register captures the Last Acknowledged Service Request (LASR) register contents when an ECC error was detected by the ICU. The ECR shows always the last ECR contents where an ECC error was detected. Software can clear the ECR contents by writing to the ECR. Error Status (STAT) and Error Overflow (EOV) bits can be used as error handling mechanism and indication that error information where lost.

If ECR.EOV is cleared by SW, ECR.EOV must be cleared together with ECR.STAT. If a new error is detected in parallel to the ECR.EOV clear than ECR.EOV is set again by hardware while ECR.STAT is cleared.

Note: In the current implementation the ECC code is only used for error detection. Detected errors are reported to the SMU but not corrected.

ECR
Error Capture Register **(xx_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
PN	[7:0]	rwh	**Service Request Priority Number** This bit field shows the priority number of the last service request where an error was detected. Bit field can be modified by writing to it.
ECC	[14:10]	rwh	**Service Request ECC** This bit field shows the ECC of the last service request where an error was detected. Bit field can be modified by writing to it. This bit field can be modified by. • writing to SRC[23:16] (byte write) • writing to SRC[31:16] (16-bit write)

Field	Bits	Type	Description
ID	[25:16]	rwh	**Service Request Node ID** This bit field shows the ID of the last service request where an error was detected. Bit field can be modified by writing to it
EOV	30	rwh	**Error Overflow Bit** The bit is set if an ECC error was detected by the ICU while ECR.STAT=´1´ (Error Overflow situation). 0_B No Error Overflow situation detected 1_B Error Overflow situation detected This bit can be cleared by writing with 1. Writing with 0 has no effect. If this bit is cleared, it must be cleared together with the STAT bit.
STAT	31	rwh	**Error Status Bit** The Error Status Bit is set whenever an ECC was detected by the ICU. 0_B No ECC error detected 1_B ECC error detected This bit can be cleared by writing with 1. Writing with 0 has no effect.
0	[29:26], 15, [9:8]	r	**Reserved** Read as 0; should be written with 0.

16.5 General Purpose Service Requests, Service Request Broadcast

The Interrupt Router module provides multiple groups of General Purpose Service Requests (GPSR) and a mechanism to trigger multiple Service Requests of a GPSR group in parallel, by software. The GPSR can be used as Software Interrupt.

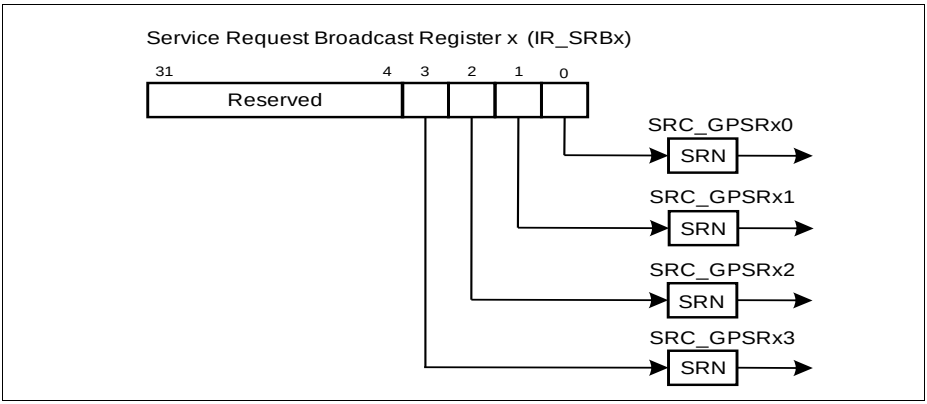


Figure 16-2 Structure of a General Purpose Service Request Group and the related Broadcast register (Example for SRC_GPSR0x, SRB0)

16.5.1 General Purpose Service Requests (GPSRxy)

The Interrupt Router module provides multiple groups of General Purpose Service Requests:

- Each General Purpose Service Request Group consists of four Service Request Nodes. Nodes that can be used as Software Interrupt
- The General Purpose Service Requests can be configured and controlled via the related Service Request Control registers SRC_GPSRxy[1)]
- The GPSR are not mapped to module service request triggers so they can only be used a SW trigger
- A General Purpose Service Request xy can only be triggered by writing ´1´ to the related SRC_GPSRxy.SETR[1)] bit or by writing a ´1´to the related Service Request Broadcast register bit SRBx[y]

16.5.2 Service Request Broadcast Registers (SRBx)

There is one Service Request Broadcast register (SRBx) implemented for each General Purpose Service Request Group (GPSRxy[1)]).

The Service Request Broadcast register x can be used to trigger multiple Service Requests within the SRC_GPSRx[1)] group in parallel.

- SRBx is always read as 0
- Writing ´1´ to SRBx[y] triggers the service request GPXRxy (y=3:0)
- Writing ´1´ to SRBx[31:4] has no effect.

1) SRC_GPSRxy: x = group number; y= number of interrupt within the group, y=0:3

16.6 System Registers

The Interrupt Router module includes the standard set of TC21x/TC22x/TC23xsystem registers. **Figure 16-3** shows these system registers which include registers for:

- Register access protection
- Module Clock Control[1)]
- Module Kernel Reset[1)]
- OCDS Control and Status Register

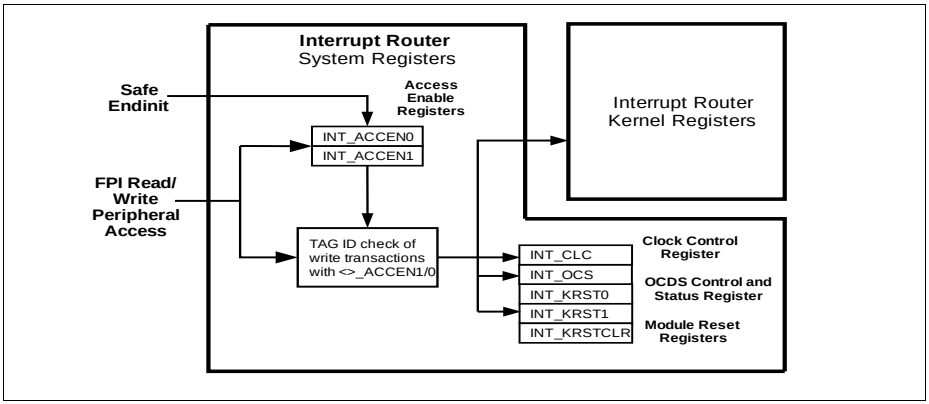


Figure 16-3 Interrupt Router System Registers

16.6.1 Register Access Protection (ACCEN1/0)

The Interrupt Router module provides a master TAG ID based write access protection as part of the AURIX safety concept. Each on chip resource with direct or indirect bus master capability has a unique master TAG ID that can be use to identify the master of an on chip bus transaction (see also chapter On Chip Bus Systems).

The SRC register is write protected via an On Chip Bus Master TAG-ID protection (see **Chapter 16.6.1**). This protection is controlled via the Interrupt Router control registers ACCEN10 and ACCEN00.

TAG ID based protection means that the support of write tansactions to the Interrupt Router control registers can be enabled / disabled for each master TAG ID individually. For a disabled master TAG ID, write access will be disconnected with error acknowledge, read access will be processed (see also **Figure 16-3**).

1) The Interrupt Router module does not support the features of this system register

The register access protection is controlled via the registers INT_ACCEN1 and INT_ACCEN0 where each bit is related to one encoding of the 6 bit On Chip Master TAG ID.

The INT_ACCEN1/0 access protection controls the write access to all Interrupt Router control and system registers with the exception of INT_ACCEN1/0. INT_ACCEN1/0 are only Safety Endinit protected.

After reset, all access enable bits and access control bits are enabled. INT_ACCEN1/0 have to be configured and checked to bring the system into a safe state.

16.6.2 Kernel Reset Registers (KRST1/0, KRSTCLR)

The Interrupt Router module does not include the kernel reset registers (KRST1, KRST0, KRSTCLR).

Note: The Interrupt Router module does not support a module kernel reset.

16.6.3 Clock Control Register (CLC)

The Interrupt Router module does not include the module clock control (CLC).

Note: The Interrupt Router module does not support the Clock Control register functionality which means that the Interrupt Router module clock can not be disabled by the CLC register.

16.6.4 OCDS Control and Status Register (OCS)

The Interrupt Router module does not include OCDS Control and Status (OCS) register.

Note: The Interrupt Router module does not support the OCS register functionality.

16.7 Arbitration Process

Each ICU in the interrupt module is related to one Interrupt Service Provider (ISP). Each ICU is related to one Interrupt Router internal arbitration bus. An arbitration bus can be related to a single ICU or it can be shared between two ICUs. This is an implementation configuration of the Interrupt Router module. If an interrupt bus is shared, the interrupt bus is arbitrating for the related ICUs in a round robin manner. Each Service Request Node (SRN) can be directed to one ISP (CPU or DMA) by mapping it via the SRC.TOS bit field setting to the related ICU.

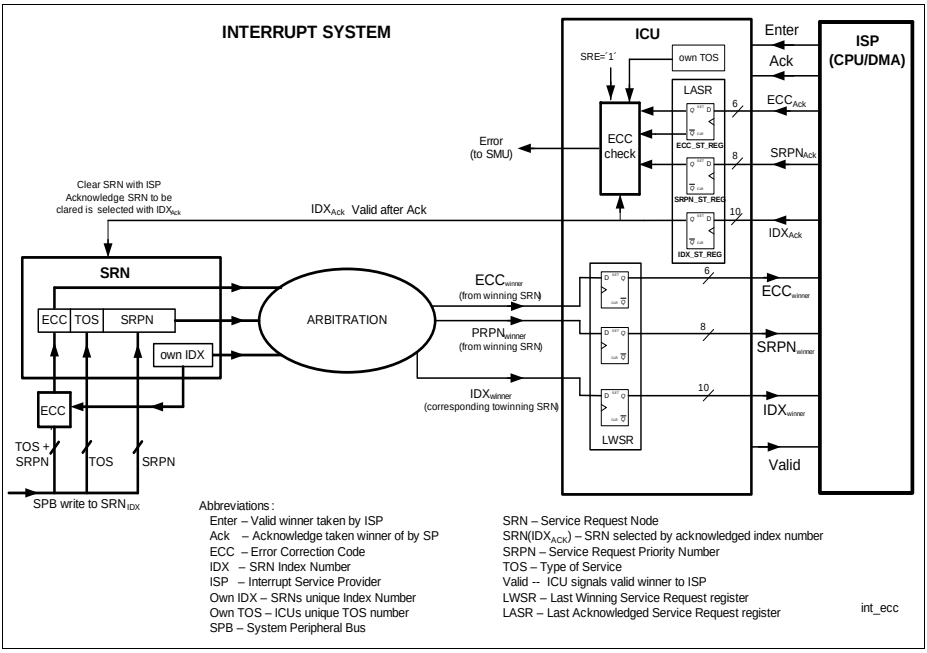


Figure 16-4 Interrupt System Arbitration Scheme Overview

With a first pending service request to an ICU, the related interrupt bus is starting an arbitration process. The arbitration process on one interrupt bus is repeated as long at least one service request mapped to the related ICUs is pending. An Interrupt Control Unit provides the service request that won the last arbitration process to its Interrupt Service Provider.

The arbitration process implemented in the TC21x/TC22x/TC23x uses 3 to 4 system peripheral bus clock cycles to determine the pending service request with the highest priority number (SRPN):

- 3 cycles as long all pending service requests are all related to one ISP. This means that the ICU can offer every 3 clock cycles a new service request to the CPU0 or to the DMA
- 4 cycles if service requests are pending for both ISPs (CPU0 and DMA). This means that the ICU can offer every 4 clock cycles a new service request to the CPU0 and every 4 clock cycles a new service request to the DMA.

During the arbitration process, the interrupt bus compares the SRC.SRPN bit fields of all pending Service Request Nodes (mapped via the SRC.TOS setting). At the end of the arbitration process, the pending service request with the highest priority number is identified as winner and the related SRN Service Request Control register bit field values SRPN, ECC and the Index of the SRN are provided to the ICU. The ICU provides these (SRPN, ECC, SRN Index) to the service provider. The ICU does an ECC check when it gets these information back from the service provider with an acknowledge. The ECC check is done with the received values: ECC, SRPN, SRN Index Number, SRE bit assumed to be´1´ (SRN enabled) and the TOS number of the ICU.

The Interrupt Router module signals detected errors to the Safety Management Unit (one bit in the SMU, covers errors from all SRN and ICUs).

Note: In the current implementation the ECC code is only used for error detection. Detected errors are reported to the SMU but not corrected.

16.7.1 Number of Clock Cycles per Arbitration Process

The Interrupt Router implementation is can be configured regarding the number of:

- supported service requests (Service Request Nodes, SRN, up to 1024)
- supported service providers (Interrupt Control Units, ICUs)
- dedicated or shared interrupt bus
- clock cycles per service request arbitration (3-4 for a shared interrupt bus, 3 SPB clock cycles for a dedicated interrupt bus)

The characteristics of the Interrupt Router module implemented in an AURIX product is described in the Module Implementation sub-chapter.

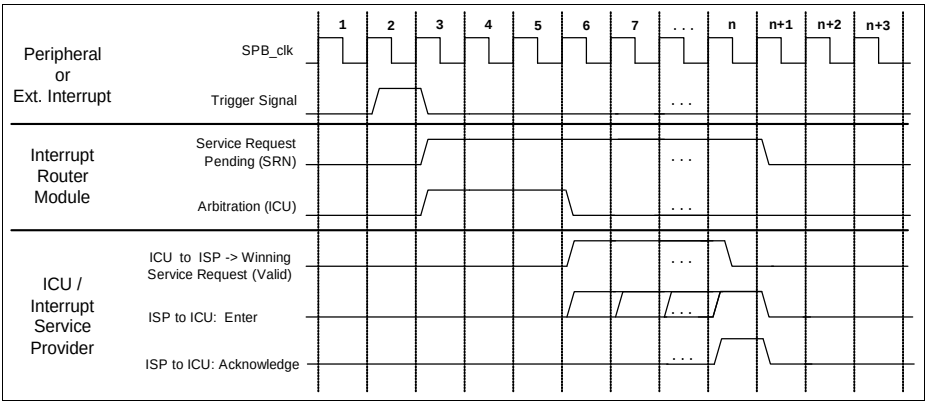


Figure 16-5 Interrupt System Timing (Schematic Overview, 3 Cycle Arbitration)

Figure 16-5 shows the interrupt timing of an Interrupt Router implementation with 2 cycle arbitration.

Cycle 1: No service request for the ICU is pending (therefore no arbitration round)

Cycle 2: One module is triggering a service request by sending a pulse to the related SRN in the Interrupt Router module.

Cycle 3-5: Arbitration among all pending service requests to the ICU

Cycle 6: ICU provides winning service request to the service provider (SRPN, ECC, SRN Index)

Cycle 7 - n-1: ICU re-arbitrates whenever a new service request from another SRN is pending, provides new winning service request to ICU if there is a new pending one with a higher SRPN number (higher priority)

Cycle 6- n-1: Interrupt Service Provider takes the information of the latest winning SRN (Enter)

Cycle n: Service provider acknowledges service request (provides SRPN, ECC, SRN Index informations of the acknowledged service request). ICU changes signals to ICU to ´no valid service request available´ in the same clock cycle.

Cycle n+1: ICU does an ECC check of the acknowledge information (SRPN, ECC, SRN Index, SRE=´1´, TOS number of the ICU. If mismatch -> Integrity Error signalled to SMU/ SRPN, ECC and Index are captured in the ECR)

Cycle n+2: ICU selects the acknowledged SRN (selected by SRN Index). Selected SRN checks acknowledged SRPN and ECC with own SRC SRPN/ECC values (if mismatch -> Integrity Error signalled to SMU)

Cycle n+3: If at least one service request to the ICU is pending: new arbitration among all pending service request to the ICU

16.7.2 Service Request Acknowledge

When a Service Provider starts with the execution of a service request that was provided by the ICU, the Service Provider sends an acknowledge to the ICU. In parallel to the acknowledge, the Service Provider sends the informations about the executed service request back to the ICU (SRPN, ECC, SRN Index Number).

In the same clock cycle, the Service Provider sends an acknowledge, the ICU changes to ´no service request available´: the ICU does not provide an arbitration winner to the service provider. This behavior of the ICU ensures that a just acknowledged service request is not provided again before the SRN was re-set (see **Figure 16-5**).

16.7.3 Handling of detected ECC Errors

The ICU does an ECC check of the informations it receives with the acknowledge from the Service Provider and the TOS number of the ICU itself. Assumption for the SRE bit is ´1´ (SRN was enabled). ECC in the SRN is covering the SRC bit field values: SRC.SRPN, SRC.SRN Index, SRC.SRE and SRC.TOS. (see **Figure 16-5**)

Note: In the current implementation the ECC code is only used for error detection. Detected errors are reported to the SMU but not corrected.

If the ICUx detects an ECC error:

- ICUx captures the ECC, SRPN and the Index that showed an ECC error in the Error capture registers (ECRx), sets the ECRx.STAT bit and the ECRx.EOV bit if the STAT bit is still set.
- ICU signals the error via the IR error signal to the TC21x/TC22x/TC23x Safety Management Unit (SMU). SMU forwards this information to a CPU (if enabled).
- ICUx clears the service request in the SRN (selected by the Index)
- The CPU that received the ´ECC error detected in Interrupt Router´ information via SMU can identify the ICU via the Error Status bits (ECRx.STAT=´1´), read out the related service request information and clear the ECRx.STAT bit by writing with ´1´. The CPU can also identify if one or multiple ECC errors where detected by this ICU via the Error Overflow bit (ECRx.EOV=´1´).

16.8 Usage of the TC21x/TC22x/TC23x Interrupt System

The following sections provide a short description of the Service Provider interfaces to the Interrupt Router ICUs.

Note: All ICU sub-modules in the Interrupt Router have the same functionality.

16.8.1 CPU to ICU Interface

Each CPU has one Interface is connected to one ICU of the Interrupt Router module. The CPU ICU interface consists of a register set where it takes over the informations of a service request provided by the ICU (SRPN, SRN Index, ECC). The informations will be send back to the ICU when the CPU acknowledges the provided service request.

The CPU ICU interface contains an Interrupt Control Register (ICR) that holds the current CPU priority number (CCPN), the global interrupt enable/disable bit (IE) and the pending interrupt priority number (PIPN). Further details of the CPU ICU interface and the CPU handling of interrupts can be found in the CPU chapter.

16.8.2 DMA to ICU Interface

The DMA module has one interface where it is connected to one ICU of the Interrupt Router module.

The DMA takes over the service request informations for the ICU, triggers internally the addressed channel and acknowledge it immediately to the ICU where the related SRN is cleared.

The DMA to ICU interface consists of a register set where it takes over the information of a service request provided by the ICU (SRPN, SRN Index, ECC), The DMA sends them back to the ICU in the next clock cycle with an acknowledge.

The DMA channel priority scheme is identical to the scheme of the interrupt system: higher SRPN number -> higher service request priority, higher DMA channel number -> higher priority of the DMA channels where channel 0 has the lowest priority within the DMA.

16.8.3 Software-Initiated Interrupts

Software can set the service request bit (SRR) in any SRN by writing to its Service Request Control Register. Thus, software can initiate service requests that are handled by the same mechanism as hardware initiated service requests.

After the SRR bit is set in an SRN, there is no way to distinguish between a software initiated service request and a hardware initiated service request. For this reason, software should only use SRNs and interrupt priority numbers that are not being used for hardware initiated service requests.

The TC21x/TC22x/TC23x contains groups of General Purpose Service Request SRNs per CPU that support software-initiated interrupts. One group per implemented TriCore CPU, each group including four SRNs. These SRNs are not connected to internal or external hardware trigger signals and can only be used as software interrupts / software initiated service requests. These SRNs are called General Purpose Service Requests Nodes (SRC_GPSRxy, x=group number, y=0-3).

Additionally, any otherwise unused SRN can be employed to generate software interrupts.

16.8.4 External Interrupts

Four SRNs (Int_SCUSRC[3:0]) are reserved to handle external interrupts. The setup for external GPIO port input signals (edge/level triggering, gating etc.) that are able to generate an interrupt request is controlled in the External Request Unit (ERU). The ERU functionality is described in detail in the SCU chapter.

16.9 Use Case Examples

This section shows a use case for the interrupt system and a use case for the OTGS.

16.9.1 Use Case Example Interrupt Handler

This section explains how to organize the TriCore interrupt vector table. When an interrupt is accepted by the TriCore, the entry address into the interrupt vector table is calculated by the base interrupt vector table pointer TriCore register BIV and the priority number of that interrupt (PIPN). The TriCore TC1.6P and TC1.6E architecture offers the possibility to configure the vector spacing per entry to either 32 Byte (see Figure below a (**Figure 16-6**) or 8 Byte (see Figure below b (**Figure 16-6**)). As a third option the vector table can be reduced to a single entry by masking the PIPN (see Figure below c (**Figure 16-6**)).

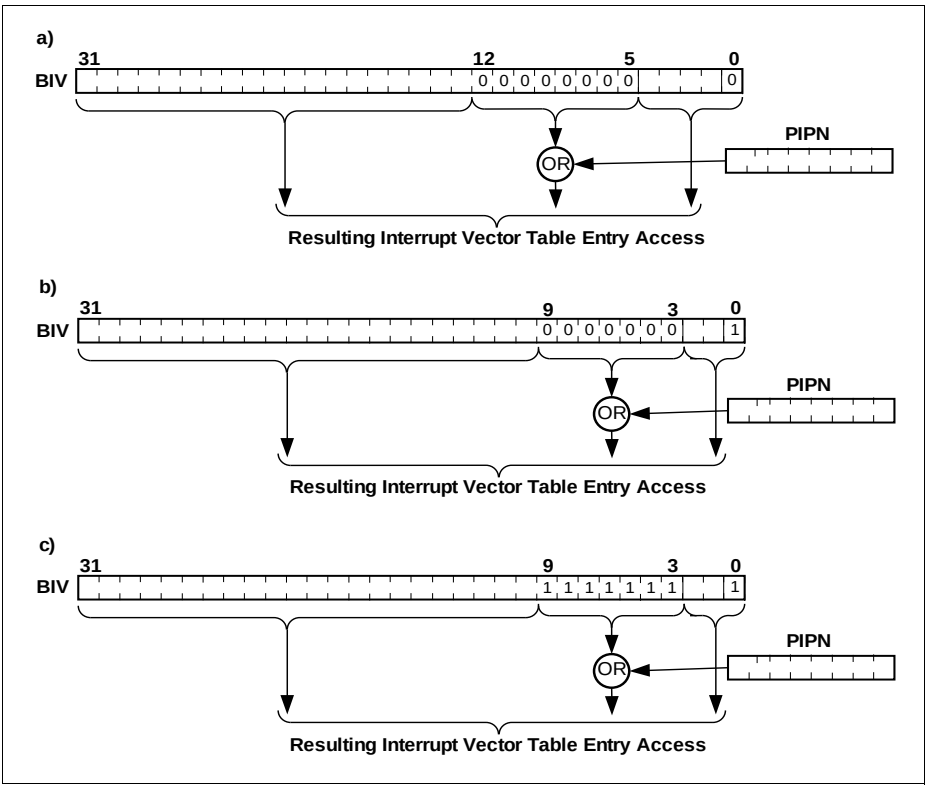


Figure 16-6 Interrupt Vector Table Address Calculation for a) 32 Byte b) 8 Byte vector entry or c) single entry

By using the 32 Byte configuration small interrupt routines can the implemented directly into the vector table. They can even span multiple vector entries (see also TriCore Architecture Manual). This type of fast interrupt handling is useful, if the vector table can

be located into the TriCore program side memory. The 8 Byte configuration reduces the vector table size. Each vector entry contains only a jump instruction or a call and return as 16-bit opcode instruction. The TriCore compiler supports this kind of interrupt vector table generation by keywords or functions. A minimum vector table can be configured if the BIV mask the PIPN so that any interrupt address calculation results in the same address. E.g.

```
__mtcr(BIV,0x80000001 | 0xFF<<3); // move to core register BIV
```

This configures the BIV register to use a common, single entry where a function interruptHandler is located to branch to the specific interrupt routine by using an array of function pointers. If a pointer to the array is used the array could be switched quickly.

Step description to initialize and install interrupts:

(Line 1) define ISR pointer array. Max. 255 interrupts possible.

(Line 2) define pointer which points to the start of the isr_pointer_array.

(Line 3) start of function interruptHandlerInstall. This function installs the interrupts in the array. Necessary information are the interrupt priority and ISR entry address.

(Line 4) This line stores the ISR entry address in the array.

(Line 5 and 6) This function branches to the specific interrupt routine and gets called immediately after an interrupt occurs.

(Line 7) This line gives the return command, after the ISR has been processed.

C Code Example to initialize and install interrupts:

```
(1) void (*isr_pointer_array[256])(void);
(2) void (**isr)(void) = isr_pointer_array;
(3) void interruptHandlerInstall(long int SRprio, long int addr){
(4) *isr_pointer_array[SRprio]=addr;
}
(5) void interruptHandler(void){
(6) isr[__mfcr(ICR) & 0xFF]();
(7) asm (" rfe");              // return from event
}
```

The interrupt entry addresses are stored in a data array instead of encoding the values into the instructions. The function interruptHandlerInstall organizes the installation of the interrupts in that array (see the application of this interrupt handler in a module e.g. use case example in STM chapter). This kind of vector table generation offers sometimes more flexibly than the 8 Byte configuration and does not require any specific compiler support for interrupts.

Note: Before an interrupt is able to occur, the interrupt system has to be globally enabled. The Interrupt Control Register (ICR) holds the global interrupt enable bit

(ICR.IE) which enables the CPU service request system. Most compiler support the attribute (or similar):
__enable();
to set this bit. (See also Architecture Manual for more details)

16.10 Module Implementation

16.10.1 Characteristics of TC21x/TC22x/TC23x Interrupt Router Module

The Interrupt Router module is implemented with the following characteristics:

- Number of clock cycles per arbitration process: 3
- Number of Interrupt Control Units (ICU): 1
- Number of General Purpose Service Request groups: 3

16.10.2 Mapping of TC21x/TC22x/TC23x Module Service Request Triggers to SRNs

All TC21x/TC22x/TC23x module service requests are mapped to one Service Request Node in the Interrupt Router.

Each SRN has one unique SRN Index Number within the Interrupt Router module.

The Interrupt Router module has one 511 interrupt trigger input vector where the interrupt trigger input vector bit [x] is related to the SRN with the SRN Index Number x[1)]. This means that a trigger pulse on interrupt trigger input vector bit [x] will trigger the SRN [x].

1) The Interrupt Router register overview table shows for all TC21x/TC22x/TC23x module service requests the SRC registers with its address offset and the related SRN Index number

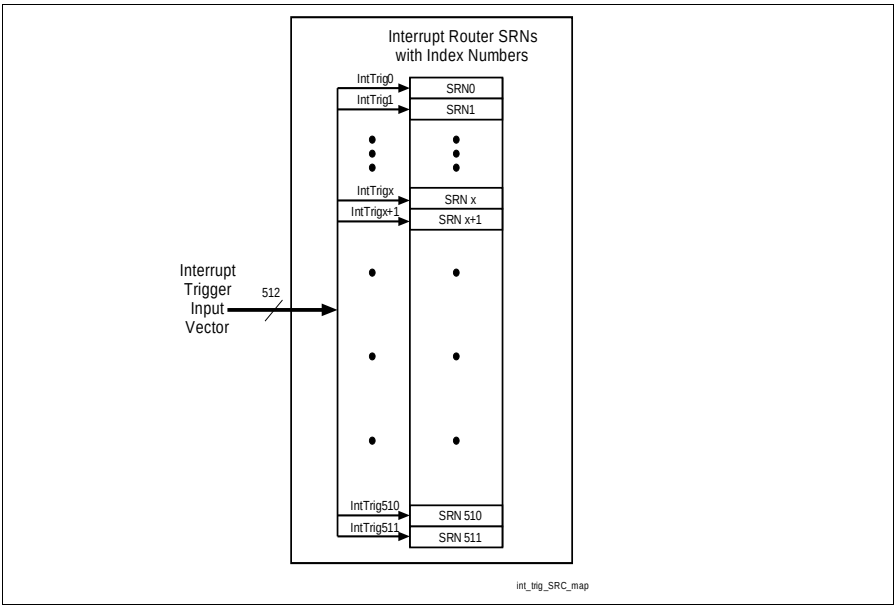


Figure 16-7 Mapping of Module Interrupt Trigger to SRN (Index Numbers)

16.10.2.1 Mapping of Service Request Control Registers

The address of SRC registers related to one module instance is identical over the whole AURIX family (e.g. interrupts of SPI0).

Each SRN has one unique SRN Index Number within the Interrupt Router module.

The number of the implemented SRNs for each AURIX device is adapted the devices feature set. All SRNs in an Interrupt Router module have consecutive Index Numbers (e.g.: if 256 SRNs are implemented, Index Numbers are 0 - 255). Each SRN has one unique SRN Index Number within the Interrupt Router module.

Example:

In a high end device with many modules/module instances SPI0 interrupts might be mapped to the SRNs with Index Numbers 180 to 185, related SRC can be accessed with address offset 0x2F0 to 0x2F5

In order to fulfill both requirements (same address for SPI0 related SRCs in all AURIX devices but reduced number of SRNs in smaller family devices with consecutive Index Numbers, the Interrupt Router module includes an address mapping shell (**Figure 16-8**). This address mapping shell re-maps the fixed SRC addresses of e.g. SPI0 of the above example to the SRN in the Interrupt Router module[1)].

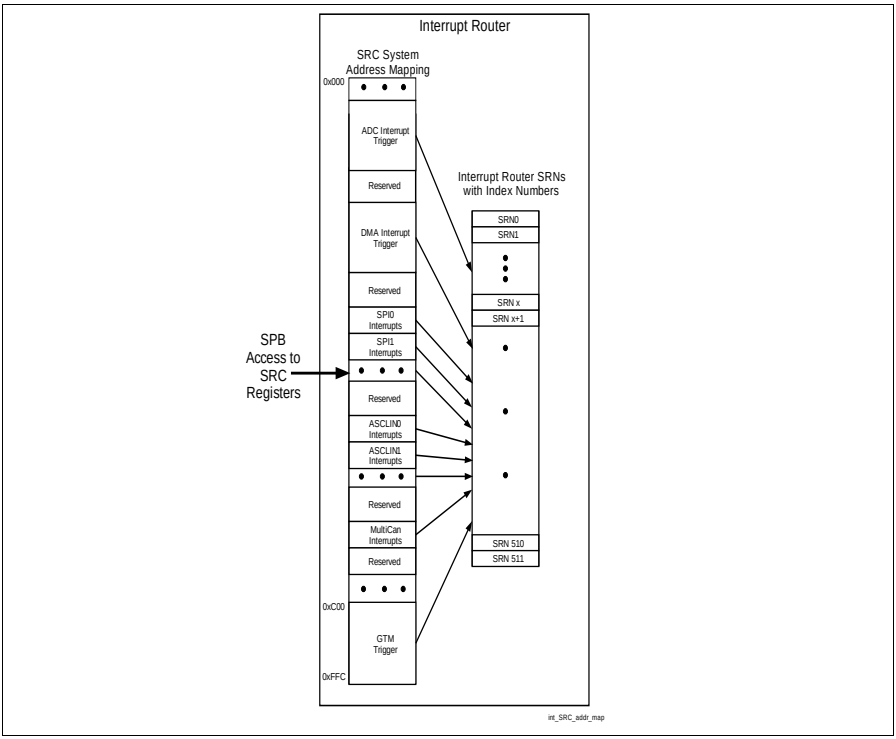


Figure 16-8 Mapping of SRC System Address to SRNs

16.10.2.2 Interrupts related to the Debug Reset

For software debug purposes the AURIX devices require some service request nodes related only to the Debug Reset. These SRNs keep its SRC register contents and the pending service request status in case of a non Debug Reset (e.g. an Application Reset). In combination with other debug reset related debug logic (e.g. breakpoint logic) this allows customer SW to debug situations that result in an application reset and after an application reset.

16.10.2.3 Timing characteristics of Service Request Trigger Signals

The Interrupt Router is clocked with the System Peripheral Bus (SPB) clock

Rules for the TC21x/TC22x/TC23x module interrupt / Service Request trigger signals to the IR:

- Trigger signals must be synchronous to SPB clock

- IR trigger inputs are edge sensitive (positive clock edge)
- Trigger signal pulse with min. high length of one SPB clock cycle, high pulse length can be > 1 SPB clock cycle
- Debug related trigger signal pulse should be kept high by the related TC21x/TC22x/TC23x module until the trigger was processed

16.11 Interrupt Router System and Module Registers

Figure 16-9 shows all registers associated with the Interrupt Router module in the TC21x/TC22x/TC23x device. The Interrupt Router allocates two address ranges:

- 2 * 256 byte address range covering the Interrupt Router system registers, ICU control registers and OTGM registers (**Table 16-1**)
- 8 KByte address range covering the Service Request Control registers (**Table 16-3**)

List of used Reset Class abbreviations

- Reset Class 1 -> Debug Reset (see description in the chapter SCU / Reset Types)
- Reset Class 3 -> Application Reset (see description in the chapter SCU / Reset Types)

List of used Access Protection Register abbreviations

- P0 -> ACCEN01/00, write protects bits [31:16] of all implemented SRC registers (SRC_xxx[31:16]).
- P1 -> ACCEN11/10, write protects bits [15:0] of all implemented SRC registers (SRC_xxx[15:0]) and the implemented ICUx Error Capture registers (INT_ECRx)

Note: A violation of the access protection will not be executed (e.g. a write to a ´Px´/ACCEN protected register from a disabled master). In this case an access protection error is signaled to the SMU. Beside this signaling to the SMU, no other error, interrupt or trap is generated.

Interrupt Router Module Registers

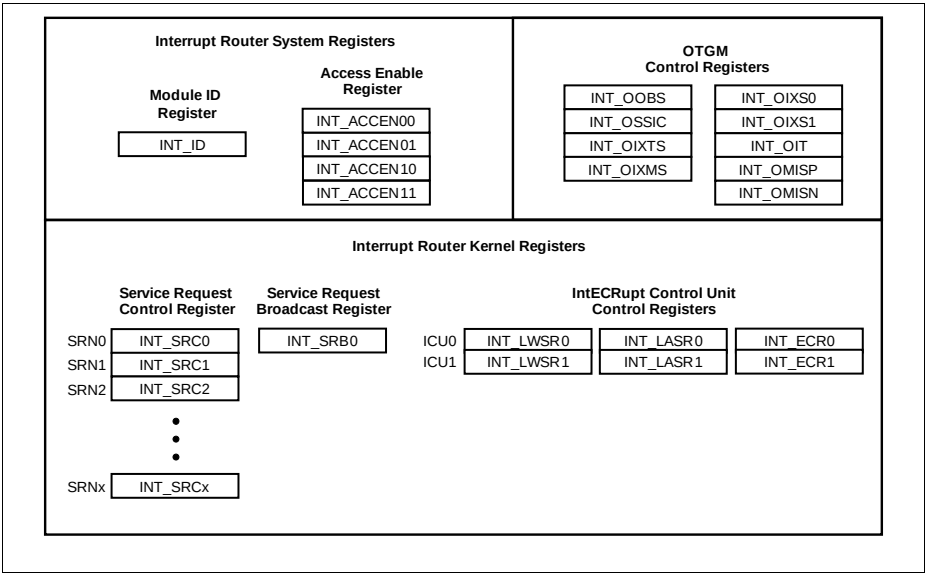


Figure 16-9 Interrupt Router module registers

Table 16-1 Registers Address Space - System, OTGM and ICU Control Registers

Module	Base Address	End Address	Note
INT	$F003\ 7000_H$	$F003\ 7FFF_H$	

Table 16-2 Registers Overview - System, OTGM and ICU Control Registers

Short Name	Description	Offset Addr.	Access Mode		Reset Class	Description See
			Read	Write		
-	Reserved	000_H-004_H	U, SV	BE	-	-
INT_ID	Interrupt Router Module Identification Register	008_H	U, SV	BE	-	Page 16-37
-	Reserved	$00C_H$	U, SV	BE	-	-

Table 16-2 Registers Overview - System, OTGM and ICU Control Registers

Short Name	Description	Offset Addr.	Access Mode		Reset Class	Description See
			Read	Write		
INT_SRB0	Service Request Broadcast Register 0	010_H	U, SV	SV, P0	3	Page 16-38
-	Reserved	014_H-$07F_H$	U, SV	BE	-	-
INT_OOBS	OTGM OTBG0/1 Status Register	080_H	U, SV	BE	1	Page 16-44
INT_OSSIC	OTGM SSI Control Register	084_H	U, SV	SV	1	Page 16-44
INT_OIXTS	OTGM IRQ MUX Trigger Set Select Register	088_H	U, SV	SV	1	Page 16-45
INT_OIXMS	OTGM IRQ MUX Missed IRQ Select Register	$08C_H$	U, SV	SV	1	Page 16-46
INT_OIXS0	OTGM IRQ MUX Select 0 Register	090_H	U, SV	SV	1	Page 16-46
INT_OIXS1	OTGM IRQ MUX Select 1 Register	094_H	U, SV	SV	1	Page 16-47
-	Reserved	098_H-$09C_H$	U, SV	BE	-	-
INT_OIT	OTGM IRQ Trace Register	$0A0_H$	U, SV	SV	1	Page 16-47
INT_OMISP	OTGM MCDS I/F Sensitivity Positive Edge Register	$0A4_H$	U, SV	SV	1	Page 16-48
INT_OMISN	OTGM MCDS I/F Sensitivity Negative Edge Register	$0A8_H$	U, SV	SV	1	Page 16-49
-	Reserved	$0AC_H$-$0EF_H$	U, SV	BE	-	-
INT_ACCEN01	Access Enable Register 1 (Access Mode, Write: P0)	$0F0_H$	U, SV	SV, SE	3	Page 16-40
INT_ACCEN00	Access Enable Register 0 (Access Mode, Write: P0)	$0F4_H$	U, SV	SV, SE	3	Page 16-39

Table 16-2 Registers Overview - System, OTGM and ICU Control Registers

Short Name	Description	Offset Addr.	Access Mode		Reset Class	Description See
			Read	Write		
INT_ACCEN11	Access Enable Register 1 (Access Mode, Write: P1)	$0F8_H$	U, SV	SV, SE	3	Page 16-40
INT_ACCEN10	Access Enable Register 0 (Access Mode, Write: P1)	$0FC_H$	U, SV	SV, SE	3	Page 16-39
INT_LWSR0	ICU0 - Last Winning Service Request Register (CPU0)	100_H	U, SV	BE	3	Page 16-40
INT_LASR0	ICU0 - Last Acknowledged Service Request Register (CPU0)	104_H	U, SV	BE	3	Page 16-41
INT_ECR0	ICU0 - Error Capture Register (CPU0)	108_H	U, SV	SV, P1	3	Page 16-42
-	Reserved	$10C_H$	U, SV	BE	-	-
INT_LWSR1	ICU1 - Last Winning Service Request Register (CPU1)	110_H	U, SV	BE	3	Page 16-40
INT_LASR1	ICU1 - Last Acknowledged Service Request Register (CPU1)	114_H	U, SV	BE	3	Page 16-41
INT_ECR1	ICU1 - Error Capture Register (CPU1)	118_H	U, SV	SV, P1	3	Page 16-42
-	Reserved	$13C_{H-FFFH}$	U, SV	BE	-	-

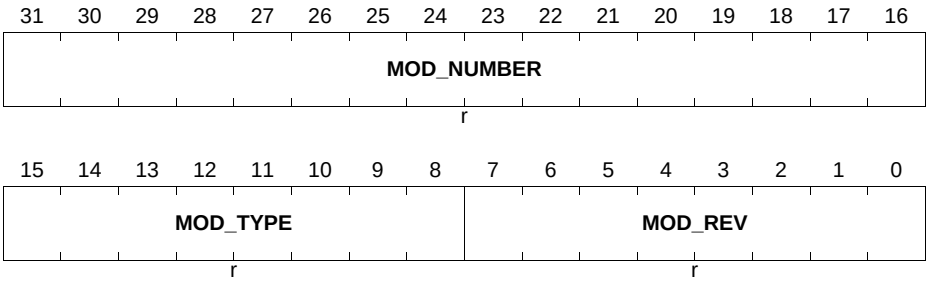
Note: Register bits marked “r” in the following register description are virtual registers and do not contain flip-flops. They are always read as 0.

16.11.1 System and ICU Control Registers

Module Identification Register (ID)

Interrupt Router Module Identification Register.

INT_ID
Module Identification Register **(008_H)** **Reset Value: 00B9 C0XX$_H$**



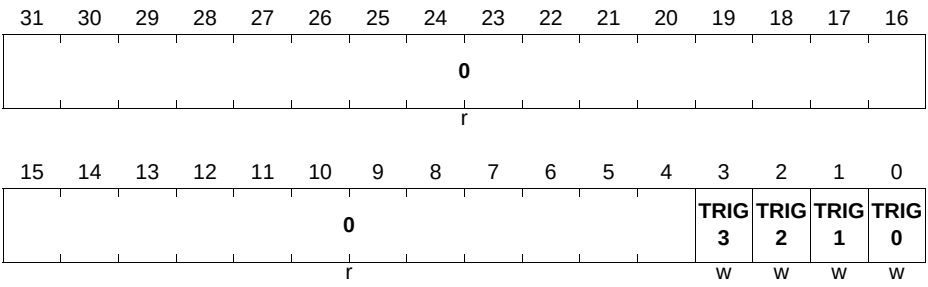
Field	Bits	Type	Description
MOD_REV	[7:0]	r	**Module Revision Number** This bit field defines the module revision number. The value of a module revision starts with 01H (first revision).
MOD_TYPE	[15:8]	r	**Module Type** The bit field is set to C0H which defines the module as a 32-bit module.
MOD_NUMBER	[31:16]	r	**Module Number Value** This bit field defines a module identification number. The value for the Interrupt Router module is 0087H.

Service Request Broadcast Register (SRB0)

Interrupt Service Request Broadcast Register can be used to trigger multiple Service Requests of a General Purpose Service Request Group in parallel.

INT_SRB0

Service Request Broadcast Register 0(010_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
TRIG0	0	w	**General Purpose Service Request Trigger 0** Writing with ´1´ SRBx.TRIG0 triggers the General Purpose Service Request 0 in group x (GPSRx0). Writing with 0 has no effect. This bit is always read as 0.
TRIG1	1	w	**General Purpose Service Request Trigger 1** Writing with ´1´ SRBx.TRIG1 triggers the General Purpose Service Request 0 in group x (GPSRx1). Writing with 0 has no effect. This bit is always read as 0.
TRIG2	2	w	**General Purpose Service Request Trigger 2** Writing with ´1´ SRBx.TRIG2 triggers the General Purpose Service Request 0 in group x (GPSRx2). Writing with 0 has no effect. This bit is always read as 0.
TRIG3	3	w	**General Purpose Service Request Trigger 3** Writing with ´1´ SRBx.TRIG3 triggers the General Purpose Service Request 0 in group x (GPSRx3). Writing with 0 has no effect. This bit is always read as 0.
0	[31:4]	r	**Reserved** Read as 0; should be written with 0.

Access Enable Register 0 (ACCEN00/10)

The Access Enable Register 0 controls write access for transactions with the on chip bus master TAG ID 000000B to 011111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The registers ACCEN00 / ACCEN01 are providing one enable bit for each 6-bit On Chip Bus Master TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000B, EN1 -> TAG ID 000001B, ... ,EN31 -> TAG ID 011111B.

INT_ACCEN00
Access Enable Register 0 ($0F4_H$) Reset Value: FFFF FFFF$_H$
INT_ACCEN10
Kernel 1 Access Enable Register 0 ($0FC_H$) Reset Value: FFFF FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN 31	**EN 30**	**EN 29**	**EN 28**	**EN 27**	**EN 26**	**EN 25**	**EN 24**	**EN 23**	**EN 22**	**EN 21**	**EN 20**	**EN 19**	**EN 18**	**EN 17**	**EN 16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN 15	**EN 14**	**EN 13**	**EN 12**	**EN 11**	**EN 10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

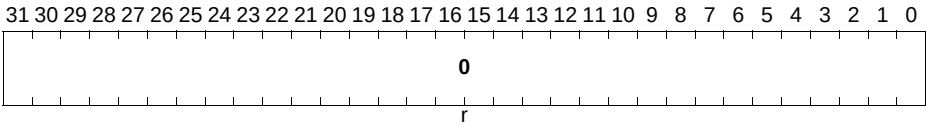
Field	Bits	Type	Description
ENn (n = 0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Access Enable Register 1 (ACCEN11/01)

The Access Enable Register 1 controls write access for transactions with the on chip bus master TAG ID 100000B to 111111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). ACCEN11/01 are not implemented with register bits as the related On Chip Bus Master TAG IDs are not used in the AURIX devices.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001B, ... ,EN31 -> TAG ID 111111B.

INT_ACCEN01
Kernel 0 Access Enable Register 1 ($0F0_H$) Reset Value: $0000\ 0000_H$
INT_ACCEN11
Kernel 1 Access Enable Register 1 ($0F8_H$) Reset Value: $0000\ 0000_H$

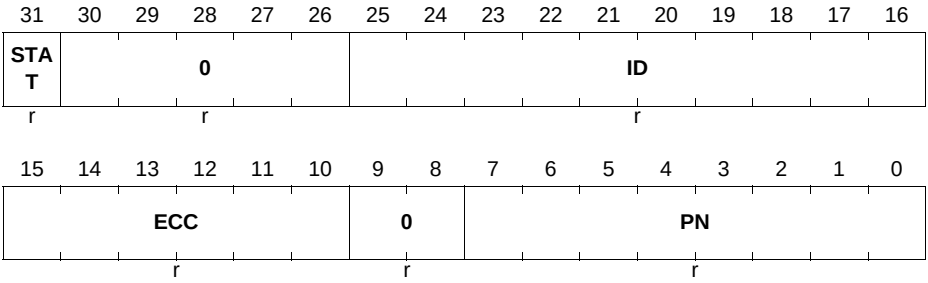


Field	Bits	Type	Description
0	[31:0]	r	**Reserved** Read as 0; should be written with 0.

Latest Winning Service Request (LWSR)

The Latest Winning Service Request register provides informations about the winner of the last arbitration round. The register bit fields are representing what is provided by the ICU to the Interrupt Service Provider.

INT_LWSRx (x = 0-1)
Latest Winning Service Request Register x (100_H+x*10_H) Reset Value: $0000\ 0000_H$



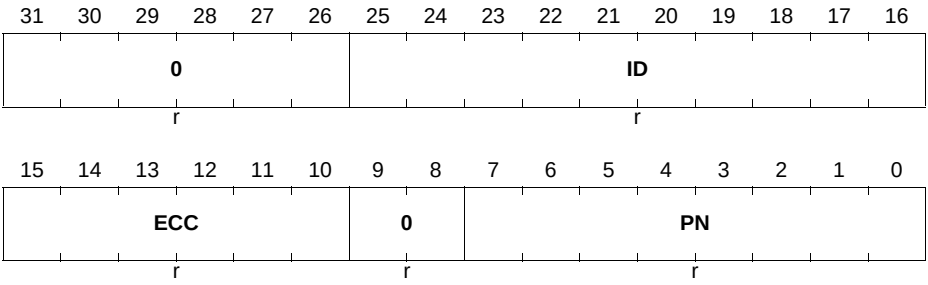
Field	Bits	Type	Description
PN	[7:0]	r	**Latest Winner Priority Number** This bit field shows the Priority Number of a pending service request that won the last arbitration round. This bit field is only valid if STAT is set to 1

Field	Bits	Type	Description
ECC	[15:10]	r	**Latest Winner ECC** This bit field shows the ECC field (SRN.ECC) that was transferred from the last winning SRN to the ICU. This bit field is only valid if STAT is set to 1.
ID	[25:16]	r	**Latest Winner Index Number** This bit field shows the index number of the last winning SRN. This bit field is only valid if STAT is set to 1
STAT	31	r	**LWSR Register Status** The STAT register indicates if the PN, ECC and ID bit fields are still valid. They are still valid if the interrupt from the SRN identified by ID is still pending. If the ICU does not have an winner because no interrupt is pending or not yet arbitrated than it clears the STAT bit. 0_B LWSR bit fields are not valid 1_B LWSR bit fields are valid
0	[30:26], [9:8]	r	**Reserved** Read as 0; should be written with 0.

Last Acknowledged Service Request (LASR)

The Last Acknowledged Service Request register provides informations about the last service request that was acknowledged by the Interrupt Service Provider. The register bit fields are representing what was sent by the Interrupt Service Provider together with the latest acknowledge.

INT_LASRx (x = 0-1)
Last Acknowledged Service Request Register x (104_H+x*10_H) **Reset Value: 0000 0000_H**

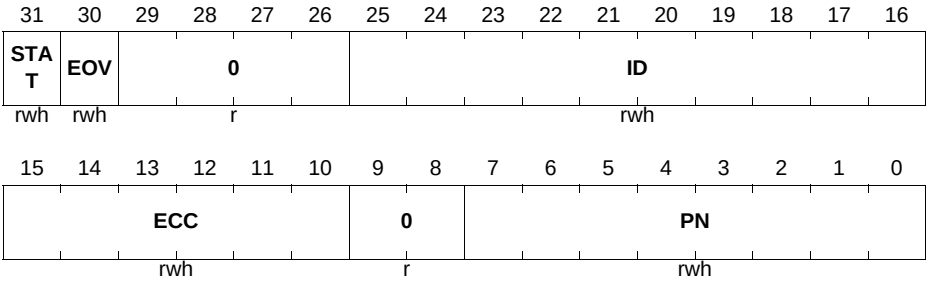


Field	Bits	Type	Description
PN	[7:0]	r	**Last Acknowledged Service Request Priority Number** This bit field shows the Priority Number of the last acknowledged service request
ECC	[15:10]	r	**Last Acknowledged Interrupt ECC** This bit field shows the ECC value of the last acknowledged service request, as send by the service provider with acknowledge
ID	[25:16]	r	**Last Acknowledged Interrupt SRN Index Number** This bit field shows the index number of the SRN, related to the last acknowledged service request, as send by the service provider with acknowledge
0	[31:26], [9:8]	r	**Reserved** Read as 0; should be written with 0.

Error Capture Register (ECR)

The Error Capture Register captures the Last Acknowledged Service Request (LASR) register contents when an ECC error was detected by the ICU. The ECR shows always the last ECR contents where an ECC error was detected. Software can clear the ECR contents by writing to the ECR.

INT_ECRx (x = 0-1)
Error Capture Register x **(108_H+x*10_H)** **Reset Value: 0000 0000_H**



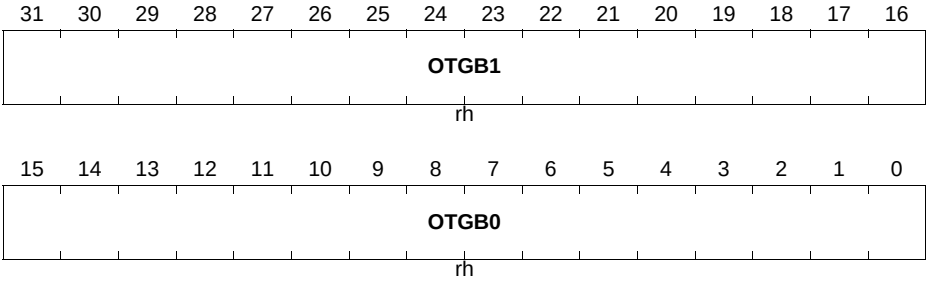
Field	Bits	Type	Description
PN	[7:0]	rwh	**Service Request Priority Number** This bit field shows the priority number of the last service request where an error was detected. Bit field can be modified by writing to it.
ECC	[15:10]	rwh	**Service Request ECC** This bit field shows the ECC of the last service request where an error was detected. Bit field can be modified by writing to it.
ID	[25:16]	rwh	**Service Request Node Index Number** This bit field shows the index number of the last service request where an error was detected. Bit field can be modified by writing to it
EOV	30	rwh	**Error Overflow Bit** The bit is set if an ECC error was detected by the ICU while ECR.STAT=´1´ (Error Overflow situation). 0_B No Error Overflow situation detected 1_B Error Overflow situation detected This bit can be cleared by writing with 1. Writing with 0 has no effect. If this bit is cleared, it must be cleared together with the STAT bit.
STAT	31	rwh	**Error Status Bit** The Error Status Bit is set whenever an ECC was detected by the ICU. 0_B No ECC error detected 1_B ECC error detected This bit can be cleared by writing with 1. Writing with 0 has no effect.
0	[29:26], [9:8]	r	**Reserved** Read as 0; should be written with 0.

16.12 OTGM Registers

All OTGM registers are cleared by Debug Reset and by each System Reset when OCDS is disabled. They are not touched by System Reset when OCDS is enabled.
Write access is 32 bit wide only and requires Supervisor Mode and OCDS enabled.

16.12.1 Status and Control

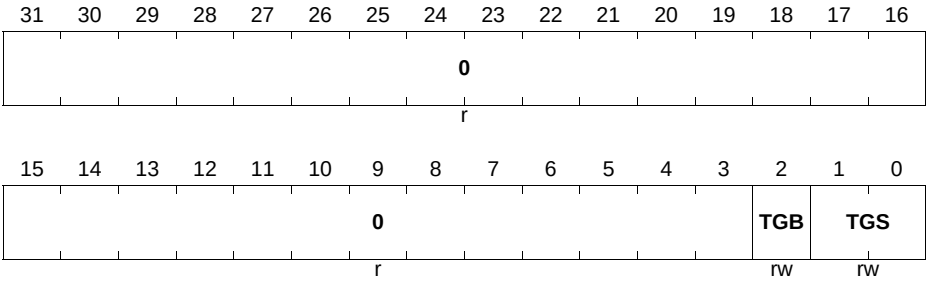
INT_OOBS
OTGM OTGB0/1 Status **(80_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
OTGB0	[15:0]	rh	**Status of OTGB0**
OTGB1	[31:16]	rh	**Status of OTGB1**

Note: OTGB0/1 value is sampled not captured. A capture register is available in OTGS.

INT_OSSIC
OTGM SSI Control **(84_H)** **Reset Value: 0000 0000$_H$**



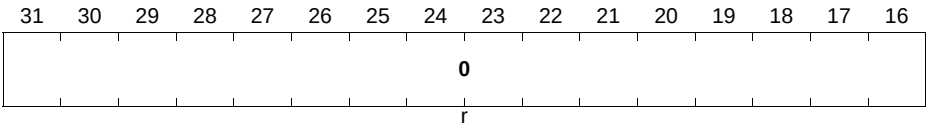
Field	Bits	Type	Description
TGS	[1:0]	rw	**Trigger Set for OTGB0/1** 0_H No Trigger Set output 1_H Trigger Set TS16_SSI (**Table 1-1**) **others**, reserved
TGB	2	rw	**OTGB0/1 Bus Select** 0_B Trigger Set is output on OTGB0 1_B Trigger Set is output on OTGB1

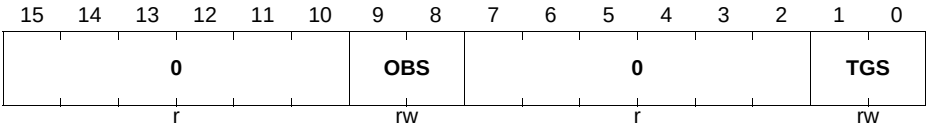
Field	Bits	Type	Description
0	[31:3]	r	**Reserved** Read as 0; must be written with 0.

16.12.2 IRQ MUX Control

INT_OIXTS

OTGM IRQ MUX Trigger Set Select (88_H) Reset Value: 0000 0000$_H$

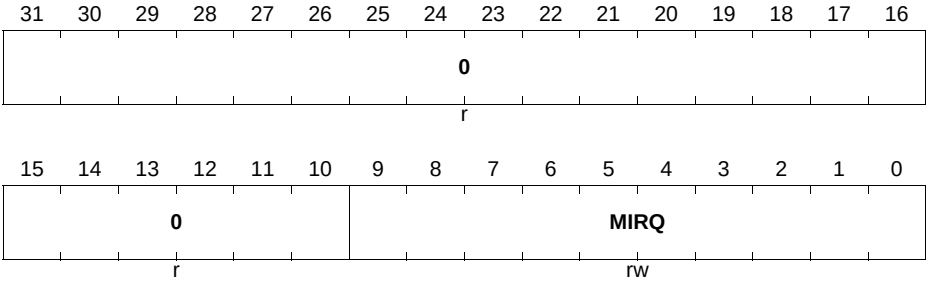




Field	Bits	Type	Description
TGS	[1:0]	rw	**Trigger Set Select for OTGB0/1 Overlay** 0_D No overlay 1_D Trigger Set TS8_IS (**Table 1-2**) 2_D Trigger Set TS8_SPA (**Table 1-3**) 3_D reserved
OBS	[9:8]	rw	**Overlay Byte Select** 0_D OTGB0 [7:0] 1_D OTGB0 [15:8] 2_D OTGB1 [7:0] 3_D OTGB1 [15:8]
0	[7:2], [31:10]	r	**Reserved** Read as 0; must be written with 0.

INT_OIXMS

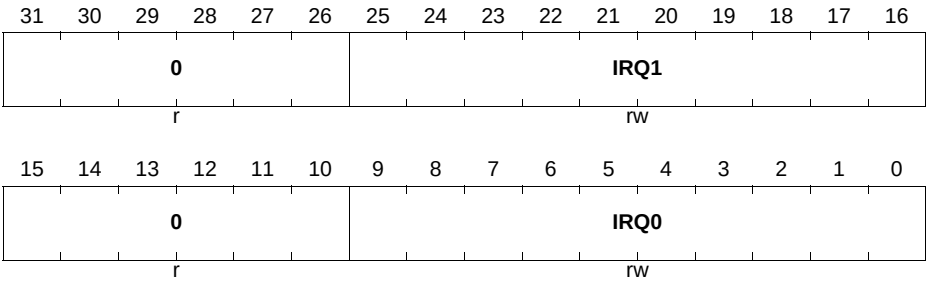
OTGM IRQ MUX Missed IRQ Select ($8C_H$) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
MIRQ	[9:0]	rw	**SRN Index for Missed Interrupt Trigger**
0	[31:10]	r	**Reserved** Read as 0; must be written with 0.

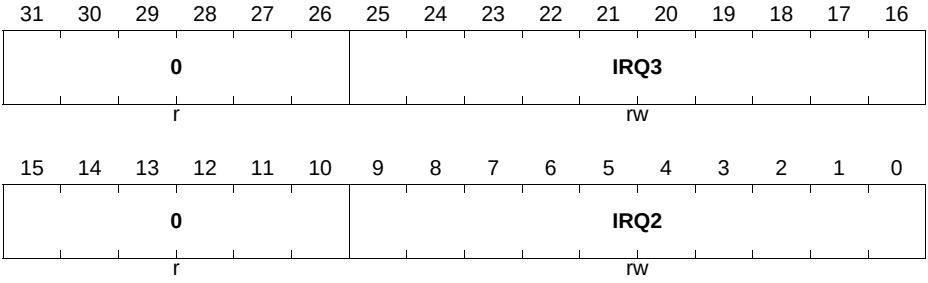
INT_OIXS0

OTGM IRQ MUX Select 0 (90_H) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
IRQ0	[9:0]	rw	**SRN Index for Interrupt Trigger 0**
IRQ1	[25:16]	rw	**SRN Index for Interrupt Trigger 1**
0	[31:26], [15:10]	r	**Reserved** Read as 0; must be written with 0.

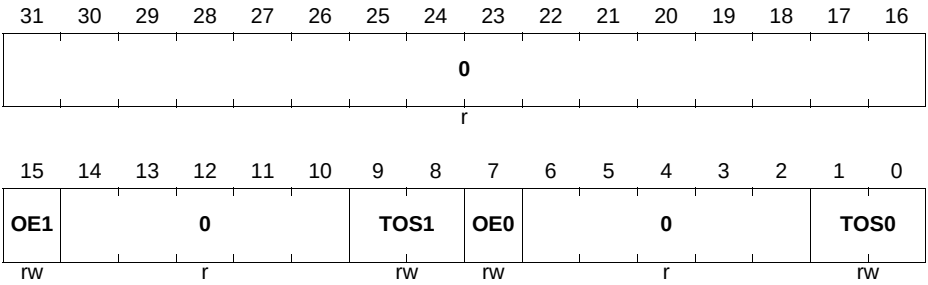
INT_OIXS1
OTGM IRQ MUX Select 1 **(94_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
IRQ2	[9:0]	rw	**SRN Index for Interrupt Trigger 2**
IRQ3	[25:16]	rw	**SRN Index for Interrupt Trigger 3**
0	[31:26], [15:10]	r	**Reserved** Read as 0; must be written with 0.

16.12.3 Interrupt System Trace

INT_OIT
OTGM IRQ Trace **($A0_H$)** **Reset Value: 0000 0000$_H$**

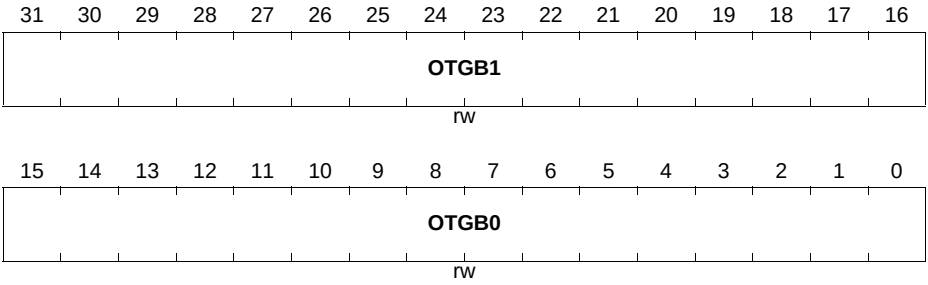


Field	Bits	Type	Description
TOS0	[1:0]	rw	**Type of Service for Observation on OTGB0** Trigger Set TS16_SP (**Table 1-4**) 0_D CPU0 service is observed 1_D DMA service is observed 2_D CPU0 service is observed 3_D DMA service is observed Compatible with SRC.TOS and with TC26/7/9x devices
OE0	7	rw	**Output Enable for OTGB0** 0_B Disabled 1_B Enabled
TOS1	[9:8]	rw	**Type of Service for Observation on OTGB1** Trigger Set TS16_SP (**Table 1-4**) 0_D CPU0 service is observed 1_D DMA service is observed 2_D CPU0 service is observed 3_D DMA service is observed Compatible with SRC.TOS and with TC26/7/9x devices
OE1	15	rw	**Output Enable for OTGB1** 0_B Disabled 1_B Enabled
0	[6:2], [14:10], [31:16]	r	**Reserved** Read as 0; must be written with 0.

16.12.4 MCDS Interface

INT_OMISP

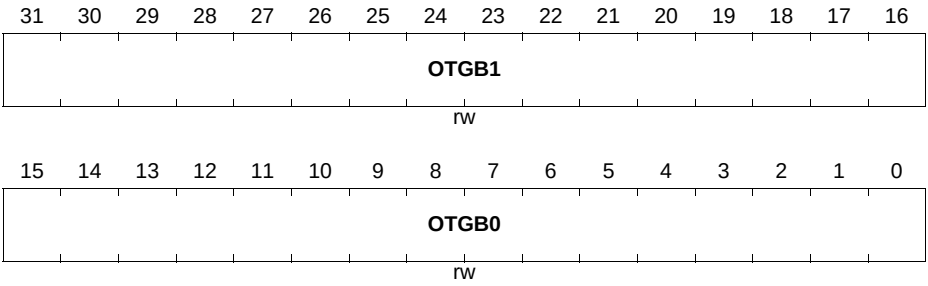
OTGM MCDS I/F Sensitivity Posedge ($A4_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
OTGB0	[15:0]	rw	**Bitwise Posedge Sensitivity for OTGB0** If a bit is set an OTGB value will be written to MCDS on a rising edge of the associated OTGB0 bit.
OTGB1	[31:16]	rw	**Bitwise Posedge Sensitivity for OTGB1** If a bit is set an OTGB value will be written to MCDS on a rising edge of the associated OTGB1 bit.

INT_OMISN

OTGM MCDS I/F Sensitivity Negedge (A8$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
OTGB0	[15:0]	rw	**Bitwise Negedge Sensitivity for OTGB0** If a bit is set an OTGB value will be written to MCDS on a falling edge of the associated OTGB0 bit.
OTGB1	[31:16]	rw	**Bitwise Negedge Sensitivity for OTGB1** If a bit is set an OTGB value will be written to MCDS on a falling edge of the associated OTGB1 bit.

16.13 Interrupt Router SRC Registers

Figure 16-9 shows all registers associated with the Interrupt Router module in the TC21x/TC22x/TC23x device. This chapter describes the Service Request Control registers including:

- Mapping of AURIX module interrupt triggers to SRC
- SRC offsets
- SRN Index numbers

List of used Reset Class abbreviations

- Reset Class 1 -> Debug Reset (see description in the chapter SCU / Reset Types)
- Reset Class 3 -> Application Reset (see description in the chapter SCU / Reset Types)

List of used Access Protection Register abbreviations

- P0 -> ACCEN01/00, write protects bits [31:16] of all implemented SRC registers (SRC_xxx[31:16]).
- P1 -> ACCEN11/10, write protects bits [15:0] of all implemented SRC registers (SRC_xxx[15:0]) and the implemented ICUx Error Capture registers (INT_ECRx)

Note: A violation of the access protection will not be executed (e.g. a write to a ´Px´/ACCEN protected register from a disabled master). In this case an access protection error is signaled to the SMU. Beside this signaling to the SMU, no other error, interrupt or trap is generated.

Table 16-3 Registers Address Space - Service Request Control Registers (SRC)

Module	Base Address	End Address	Note
SRC	$F003\ 8000_H$	$F003\ 9FFF_H$	

Table 16-4 Registers Overview - Service Request Control Registers

Short Name	Module	Description	Offset Addr.	Access Mode		Reset Class	Index NR.
				Read	Write		
SRC_CPU0 SBSRC	CPU0	CPU 0 Software Breakpoint Service Request	0000_H	U, SV	SV, P0, P1	1	0
-	-	Reserved	0004_H-$001C_H$	U, SV	BE	-	-

Table 16-4 Registers Overview - Service Request Control Registers

Short Name	Module	Description	Offset Addr.	Access Mode		Reset Class	Index NR.
				Read	Write		
SRC_EMEM	EMEM	Emulation Memory Service Request (on ED devices only)	0020_H	U, SV	SV, P0, P1	1	1
-	-	Reserved	0024_H-$003C_H$	U, SV	BE	-	-
SRC_BCUSPBSBSRC	BCU	Bus Control Unit SPB Service Request	0040_H	U, SV	SV, P0, P1	1	2
-	-	Reserved	0044_H	U, SV	BE	-	-
SRC_XBARSRC	XBAR_SRI	XBAR_SRI Service Request	0048_H	U, SV	SV, P0, P1	1	3
-	-	Reserved	$004C_H$	U, SV	BE	-	-
SRC_Cerberusm	CERBERUS	Cerberus Service Request m (m = 0-1)	0050_H+ (m × 4_H)	U, SV	SV, P0, P1	1	4 + m
-	-	Reserved	0058_H-$007C_H$	U, SV	BE	-	-
SRC_ASCLINmTX	ASCLIm	ASCLIN m Transmit Service Request (m = 0-1)	0080_H+ (m × C_H)	U, SV	SV, P0, P1	3	6 + m*3
SRC_ASCLINmRX	ASCLIm	ASCLIN m Receive Service Request (m = 0-1)	0084_H+ (m × C_H)	U, SV	SV, P0, P1	3	7 + m*3
SRC_ASCLINmEX	ASCLIm	ASCLIN m Error Service Request (m = 0-1)	0088_H+ (m × C_H)	U, SV	SV, P0, P1	3	8 + m*3
-	-	Reserved	$00B0_H$-$018C_H$	BE	BE	-	-

Table 16-4 Registers Overview - Service Request Control Registers

Short Name	Module	Description	Offset Addr.	Access Mode		Reset Class	Index NR.
				Read	Write		
SRC_QSPImTX	QSPIm	QSPI m Transmit Service Request (m = 0-3	0190_H+ (m × 18_H)	U, SV	SV, P0, P1	3	12 + m*6
SRC_QSPImRX	QSPIm	QSPI m Receive Service Request (m = 0-3)	0194_H+ (m × 18_H)	U, SV	SV, P0, P1	3	13 + m*6
SRC_QSPImERR	QSPIm	QSPI m Error Service Request (m = 0-3)	0198_H+ (m × 18_H)	U, SV	SV, P0, P1	3	14 + m*6
SRC_QSPImPT	QSPIm	QSPI m Phase Transition Service Request (m = 0-3)	$019C_H$+ (m × 18_H)	U, SV	SV, P0, P1	3	15 + m*6
SRC_RESERVED1m	RESERVED	Reserved Service Request 1m (m = 0-1)	$01A0_H$+ (m × 18_H)	U, SV	SV, P0, P1	3	16 + m*6
SRC_QSPImHC	QSPIm	QSPI m High Speed Capture Service Request (m = 2-3)	$01A0_H$+ (m × 18_H)	U, SV	SV, P0, P1	3	16 + m*6
SRC_QSPImU	QSPIm	QSPI m User Defined Service Request (m = 0-3)	$01A4_H$+ (m × 18_H)	U, SV	SV, P0, P1	3	17 + m*6
-	-	Reserved	0208_H-$034C_H$	U, SV	BE	-	-
SRC_SENTm	SENT	SENT TRIGm Service Request (m = 0-3)	0350_H+ (m × 4_H)	U, SV	SV, P0, P1	3	36 + m
-	-	Reserved	0360_H-$041C_H$	BE	BE	-	-
SRC_CCU6mSR0	CCU6m	CCU6 m Service Request 0 (m = 0-1)	0420_H+ (m × 10_H)	U, SV	SV, P0, P1	3	40 + m*4

Table 16-4 Registers Overview - Service Request Control Registers

Short Name	Module	Description	Offset Addr.	Access Mode		Reset Class	Index NR.
				Read	Write		
SRC_CCU6mSR1	CCU6m	CCU6 m Service Request 1 (m = 0-1)	0424_H+ (m × 10_H)	U, SV	SV, P0, P1	3	41 + m*4
SRC_CCU6mSR2	CCU6m	CCU6 m Service Request 2 (m = 0-1)	0428_H+ (m × 10_H)	U, SV	SV, P0, P1	3	42 + m*4
SRC_CCU6mSR3	CCU6m	CCU6 m Service Request 3 (m = 0-1)	$042C_H$+ (m × 10_H)	U, SV	SV, P0, P1	3	43 + m*4
SRC_GPT120CIRQ	GPT120	GPT120 CAPREL Service Request	0460_H	U, SV	SV, P0, P1	3	48
SRC_GPT120T2	GPT120	GPT120 T2 Overflow/Underflow Service Request	0464_H	U, SV	SV, P0, P1	3	49
SRC_GPT120T3	GPT120	GPT120 T3 Overflow/Underflow Service Request	0468_H	U, SV	SV, P0, P1	3	50
SRC_GPT120T4	GPT120	GPT120 T4 Overflow/Underflow Service Request	$046C_H$	U, SV	SV, P0, P1	3	51
SRC_GPT120T5	GPT120	GPT120 T5 Overflow/Underflow Service Request	0470_H	U, SV	SV, P0, P1	3	52
SRC_GPT120T6	GPT120	GPT120 T6 Overflow/Underflow Service Request	0474_H	U, SV	SV, P0, P1	3	53
SRC_STM0SR0	STM0	System Timer 0 Service Request 0	0490_H	U, SV	SV, P0, P1	3	54
SRC_STM0SR1	STM0	System Timer 0 Service Request 1	0494_H	U, SV	SV, P0, P1	3	55

Table 16-4 Registers Overview - Service Request Control Registers

Short Name	Module	Description	Offset Addr.	Access Mode		Reset Class	Index NR.
				Read	Write		
-	-	Reserved	$04A0_H$-$04EC_H$	BE	BE	-	-
SRC_DMAERR	DMA	DMA Error Service Request	04F0	U, SV	SV, P0, P1	3	56
-	-	Reserved	$04F4_H$-$04FC_H$	BE	BE	3	-
SRC_DMACHm	DMA	DMA Channel m Service Request (m = 0-15)	0500_H+ (m × 4_H)	U, SV	SV, P0, P1	3	57 + m
-	-	Reserved	0540_H-$08EC_H$	BE	BE	-	-
SRC_ETH	ETH	Ethernet Service Request	$08F0_H$	U, SV	SV, P0, P1	3	73
-	-	Reserved	$08F4_H$-$08FC_H$	BE	BE	-	-
SRC_CANINTm	MultiCAN	MultiCAN Service Request m (m = 0-15)	0900_H+ (m × 4_H)	U, SV	SV, P0, P1	3	74 + m
SRC_CAN1INTm	MultiCAN1	MultiCAN1 Service Request m (m = 0-7)	0940_H+ (m × 4_H)	U, SV	SV, P0, P1	3	90+ m
-	-	Reserved	0960_H-$097C_H$	BE	BE	-	-
SRC_VADCG0SRm	VADC	VADC Group 0 Service Request m (m = 0-3)	0980_H+ (m × 4_H)	U, SV	SV, P0, P1	3	98 + m
SRC_VADCG1SRm	VADC	VADC Group 1 Service Request m (m = 0-3)	0990_H+ (m × 4_H)	U, SV	SV, P0, P1	3	102 + m

Table 16-4 Registers Overview - Service Request Control Registers

Short Name	Module	Description	Offset Addr.	Access Mode		Reset Class	Index NR.
				Read	Write		
SRC_VADCG2SRm	VADC	VADC Group 2 Service Request m (m = 0-3)	$09A0_H$+ (m × 4_H)	U, SV	SV, P0, P1	3	106 + m
SRC_VADCG3SRm	VADC	VADC Group 3 Service Request m (m = 0-3)	$09B0_H$+ (m × 4_H)	U, SV	SV, P0, P1	3	110 + m
-	-	Reserved	$09C0_H$-$0A9C_H$	BE	BE	-	-
SRC_VADCCG0SRm	VADC	VADC Common Group 0 Service Request m (m = 0-3)	$0AA0_H$+ (m × 4_H)	U, SV	SV, P0, P1	3	114 + m
-	-	Reserved	$0AB0_H$-$0BDC_H$	BE	BE	-	-
SRC_ERAYINT0	E-RAY	E-RAY Service Request 0	$0BE0_H$	U, SV	SV, P0, P1	3	118
SRC_ERAYINT1	E-RAY	E-RAY Service Request 1	$0BE4_H$	U, SV	SV, P0, P1	3	119
SRC_ERAYTINT0	E-RAY	E-RAY Timer Interrupt 0 Service Request	$0BE8_H$	U, SV	SV, P0, P1	3	120
SRC_ERAYTINT1	E-RAY	E-RAY Timer Interrupt 1 Service Request	$0BEC_H$	U, SV	SV, P0, P1	3	121
SRC_ERAYNDAT0	E-RAY	E-RAY New Data 0 Service Request	$0BF0_H$	U, SV	SV, P0, P1	3	122
SRC_ERAYNDAT1	E-RAY	E-RAY New Data 1 Service Request	$0BF4_H$	U, SV	SV, P0, P1	3	123

Table 16-4 Registers Overview - Service Request Control Registers

Short Name	Module	Description	Offset Addr.	Access Mode		Reset Class	Index NR.
				Read	Write		
SRC_ERAYMBSC0	E-RAY	E-RAY Message Buffer Status Changed 0 Service Request	$0BF8_H$	U, SV	SV, P0, P1	3	124
SRC_ERAYMBSC1	E-RAY	E-RAY Message Buffer Status Changed 1 Service Request	$0BFC_H$	U, SV	SV, P0, P1	3	125
SRC_ERAYOBUSY	E-RAY	E-RAY Output Buffer Busy Service Request	$0C00_H$	U, SV	SV, P0, P1	3	126
SRC_ERAYIBUSY	E-RAY	E-RAY Input Buffer Busy Service Request	$0C04_H$	U, SV	SV, P0, P1	3	127
-	-	Reserved	$0C08_H$-$0C2C_H$	BE	BE	-	-
SRC_PMU00	PMU0	PMU 0 Service Request 0	$0C30_H$	U, SV	SV, P0, P1	3	128
SRC_PMU01	PMU0	PMU 0 Service Request 1	$0C34_H$	U, SV	SV, P0, P1	3	129
-	-	Reserved	$0C38_H$-$0CBC_H$	BE	BE	-	-
SRC_HSMm	HSM	TC23x: HSM Service Request x (m = 0-1) TC22x: Reserved	$0CC0_H$+(m × 4_H)	U, SV	SV, P0, P1	3	130 + m
-	-	Reserved	$0CC8_H$-$0CCC_H$	BE	BE	-	-
SRC_SCUDTS	SCU	SCU DTS Busy Service Request	$0CD0_H$	U, SV	SV, P0, P1	3	132

Table 16-4 Registers Overview - Service Request Control Registers

Short Name	Module	Description	Offset Addr.	Access Mode		Reset Class	Index NR.
				Read	Write		
SRC_SCUERUm	SCU	SCU ERU Service Request x (m = 0-3)	$0CD4_H$+ (m × 4_H)	U, SV	SV, P0, P1	3	133 + m
-	-	Reserved	$0CE4_H$-$0D0C_H$	BE	BE	-	-
SRC_SMUm	SMU	SMU Service Request m (m = 0-2)	$0D10_H$+ (m × 4_H)	U, SV	SV, P0, P1	3	137 + m
-	-	Reserved	$0D1C_H$-$0DDC_H$	BE	BE	-	-
SRC_LMU	LMU	LMU Error Service Request	$0DE0_H$	U, SV	SV, P0, P1	3	140
-	-	Reserved	$0DE4_H$-$0FAC_H$	BE	BE	-	-
SRC_EVRWUT	EVR	EVR Wake Up Timer Service Request	$0FB0_H$	U, SV	SV, P0, P1	3	141
SRC_SCDC	EVR	EVR Service Request	$0FB4_H$	U, SV	SV, P0, P1	3	142
-	-	Reserved	$0FB8_H$-$0FBC_H$	BE	BE	-	-
SRC_FFTDONE	-	TC23x: FFT Done Service Request TC22x: Reservedt	$0FC0_H$	U, SV	SV, P0, P1	3	143
SRC_FFTERR	-	TC23x:FFT Error Service Request TC22x: Reserved	$0FC4_H$	U, SV	SV, P0, P1	3	144
SRC_FFTRFS	-	TC23x:FFT Ready For Start Service Request TC22x: Reserved	$0FC8_H$	U, SV	SV, P0, P1	3	145

Table 16-4 Registers Overview - Service Request Control Registers

Short Name	Module	Description	Offset Addr.	Access Mode		Reset Class	Index NR.
				Read	Write		
-	-	Reserved	$0FCC_H$-$0FFC_H$	BE	BE	-	-
SRC_GPSR0m	IR	General Purpose Service Request 0 m (m = 0-3)	1000_H+(m × 4_H)	U, SV	SV, P0, P1	3	146 + m
-	-	Reserved	1010_H-$15FC_H$	BE	BE	-	-
SRC_GTMAEIIRQ	GTM	AEI Shared Service Request	1600_H	U, SV	SV, P0, P1	3	150
-	-	Reserved	1604_H-$176C_H$	BE	BE	-	-
SRC_GTMERR	GTM	Error Service Request	1770_H	U, SV	SV, P0, P1	3	151
-	-	Reserved	1774_H-$177C_H$	BE	BE	-	-
SRC_GTMTIM0m	GTM	TIM0 Shared Service Request m (m = 0-7)	1780_H+(m × 4_H)	U, SV	SV, P0, P1	3	152 + m
-	-	Reserved	$17A0_H$-$1B7C_H$	BE	BE	-	-
SRC_GTMTOM0m	GTM	TOM0 Shared Service Request m (m = 0-7)	$1B80_H$+(m × 4_H)	U, SV	SV, P0, P1	3	160 + m
SRC_GTMTOM1m	GTM	TOM1 Shared Service Request m (m = 0-7)	$1BA0_H$+(m × 4_H)	U, SV	SV, P0, P1	3	168 + m
-	-	Reserved	$1BC0_H$-$1FFC_H$	BE	BE	-	-

Note: Register bits marked “r” in the following register description are virtual registers and do not contain flip-flops. They are always read as 0.

SRC_CPU0SBSRC
CPU 0Software Breakpoint Service Request
(0000_H) **Reset Value: 0000 0000_H**

SRC_EMEM
Emulation Memory Service Request (0020_H) **Reset Value: 0000 0000_H**

SRC_BCUSPBSBSRC
Bus Control Unit SPB Service Request
(0040_H) **Reset Value: 0000 0000_H**

SRC_XBARSRC
XBAR_SRI Service Request
(0048_H) **Reset Value: 0000 0000_H**

SRC_CERBERUSm (m=0-1)
Cerberus Service Request m
(0050_H+m*04_H) **Reset Value: 0000 0000_H**

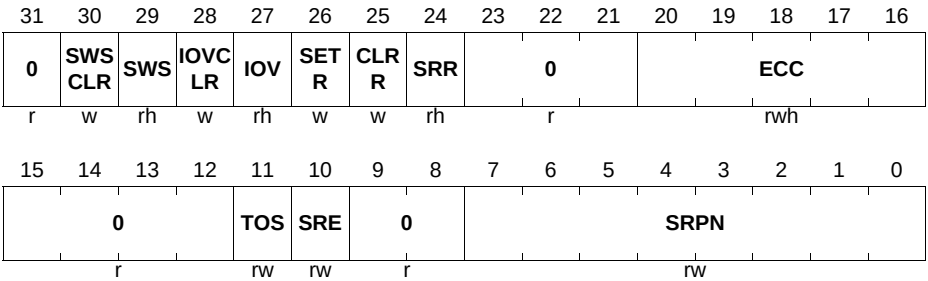
SRC_ASCLINmTX (m=0-1)
ASCLIN m Transmit Service Request
(0080_H+m*$0C_H$) **Reset Value: 0000 0000_H**

SRC_ASCLINmRX (m=0-1)
ASCLIN m Receive Service Request
(0084_H+m*$0C_H$) **Reset Value: 0000 0000_H**

SRC_ASCLINmERR (m=0-1)
ASCLIN m Error Service Request
(0088_H+m*$0C_H$) **Reset Value: 0000 0000_H**

SRC_QSPImTX (m=0-3)
QSPI m Transmit Service Request
(0190_H+m*18_H) **Reset Value: 0000 0000_H**

SRC_QSPImRX (m=0-3)
QSPI m Receive Service Request
(0194_H+m*18_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number** 00_H Service request is on lowest priority 01_H Service request is one before lowest priority ... ... FF_H Service request is on highest priority *Note: For a CPU 01H is the lowest priority as 00H is never serviced. For the DMA 00H triggers channel 0*
SRE	10	rw	**Service Request Enable** 0_B Service request is disabled 1_B Service request is enabled
TOS	11	rw	**Type of Service Control** 0_H CPU0 service is initiated 1_H DMA service is initiated
ECC	[20:16]	rwh	**ECC**
SRR	24	rh	**Service Request Flag** 0_B No service request is pending 1_B A service request is pending
CLRR	25	w	**Request Clear Bit** 0_B No action 1_B Clear SRR; bit value is not stored; read always returns 0; no action if SETR is set in parallel.
SETR	26	w	**Request Set Bit** 0_B No action 1_B Set SRR and SWS; SRR bit value is not stored; read always returns 0; no action if CLRR is set in parallel.
IOV	27	rh	**Interrupt Trigger Overflow Bit** 0_B No Interrupt Trigger Overflow detected 1_B Interrupt Overflow Detected.
IOVCLR	28	w	**Interrupt Trigger Overflow Clear Bit** 0_B No action 1_B Clear IOV; bit value is not stored; read always returns 0.

Field	Bits	Type	Description
SWS	29	rh	**SW Sticky Bit** 0_B No interrupt was initiated via SETR 1_B Interrupt was initiated via SETR
SWSCLR	30	w	**SW Sticky Clear Bit** 0_B No action 1_B Clear SWS; bit value is not stored; read always returns 0.
0	31, [23:21], [15:12], [9:8]	r	**Reserved** Read as 0; should be written with 0.

SRC_QSPImERR (m=0-3)
QSPI m Error Service Request
(0198_H+m*18_H) **Reset Value: 0000 0000_H**

SRC_QSPImPT (m=0-3)
QSPI m Phase Transition Service Request
($019C_H$+m*18_H) **Reset Value: 0000 0000_H**

SRC_RESERVED1m (m=0-1)
Reserved Service Request 1m
($01A0_H$+m*18_H) **Reset Value: 0000 0000_H**

SRC_QSPImHC (m=2-3)
QSPI m High Speed Capture Service Request
($01A0_H$+m*18_H) **Reset Value: 0000 0000_H**

SRC_QSPImU (m=0-3)
QSPI m User Defined Service Request
($01A4_H$+m*18_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**SWSCLR**	**SWS**	**IOVCLR**	**IOV**	**SETR**	**CLRR**	**SRR**	**0**			**ECC**				
r	w	rh	w	rh	w	w	rh	r			rwh				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				**TOS**	**SRE**	**0**		**SRPN**							
r				rw	rw	r		rw							

Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number** 00_H Service request is on lowest priority 01_H Service request is one before lowest priority ... ... FF_H Service request is on highest priority *Note: For a CPU 01H is the lowest priority as 00H is never serviced. For the DMA 00H triggers channel 0*
SRE	10	rw	**Service Request Enable** 0_B Service request is disabled 1_B Service request is enabled
TOS	11	rw	**Type of Service Control** 0_H CPU0 service is initiated 1_H DMA service is initiated
ECC	[20:16]	rwh	**ECC**
SRR	24	rh	**Service Request Flag** 0_B No service request is pending 1_B A service request is pending
CLRR	25	w	**Request Clear Bit** 0_B No action 1_B Clear SRR; bit value is not stored; read always returns 0; no action if SETR is set in parallel.
SETR	26	w	**Request Set Bit** 0_B No action 1_B Set SRR and SWS; SRR bit value is not stored; read always returns 0; no action if CLRR is set in parallel.
IOV	27	rh	**Interrupt Trigger Overflow Bit** 0_B No Interrupt Trigger Overflow detected 1_B Interrupt Overflow Detected.
IOVCLR	28	w	**Interrupt Trigger Overflow Clear Bit** 0_B No action 1_B Clear IOV; bit value is not stored; read always returns 0.

Field	Bits	Type	Description
SWS	29	rh	**SW Sticky Bit** 0_B No interrupt was initiated via SETR 1_B Interrupt was initiated via SETR
SWSCLR	30	w	**SW Sticky Clear Bit** 0_B No action 1_B Clear SWS; bit value is not stored; read always returns 0.
0	31, [23:21], [15:12], [9:8]	r	**Reserved** Read as 0; should be written with 0.

SRC_SENTm (m=0-3)
SENT TRIGm Service Request **(0350_H+m*04_H)** **Reset Value: 0000 0000_H**
SRC_CCU6mSR0 (m=0-1)
CCU6 m Service Request 0 **(0420_H+m*10_H)** **Reset Value: 0000 0000_H**
SRC_CCU6mSR1 (m=0-1)
CCU6 m Service Request 1 **(0424_H+m*10_H)** **Reset Value: 0000 0000_H**
SRC_CCU6mSR2 (m=0-1)
CCU6 m Service Request 2 **(0428_H+m*10_H)** **Reset Value: 0000 0000_H**
SRC_CCU6mSR3 (m=0-1)
CCU6 m Service Request 3 **($042C_H$+m*10_H)** **Reset Value: 0000 0000_H**
SRC_GPT120CIRQ
GPT120 CAPREL Service Request **(0460_H)** **Reset Value: 0000 0000_H**
SRC_GPT120T2
GPT120 T2 Overflow/Underflow Service Request
(0464_H) **Reset Value: 0000 0000_H**
SRC_GPT120T3
GPT120 T3 Overflow/Underflow Service Request
(0468_H) **Reset Value: 0000 0000_H**
SRC_GPT120T4
GPT120 T4 Overflow/Underflow Service Request
($046C_H$) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	SWSCLR	SWS	IOVCLR	IOV	SETR	CLRR	SRR	0			ECC				
r	w	rh	w	rh	w	w	rh	r			rwh				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				TOS	SRE	0		SRPN							
r				rw	rw	r		rw							

Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number** 00_H Service request is on lowest priority 01_H Service request is one before lowest priority ... ... FF_H Service request is on highest priority *Note: For a CPU 01H is the lowest priority as 00H is never serviced. For the DMA 00H triggers channel 0*
SRE	10	rw	**Service Request Enable** 0_B Service request is disabled 1_B Service request is enabled
TOS	11	rw	**Type of Service Control** 0_H CPU0 service is initiated 1_H DMA service is initiated
ECC	[20:16]	rwh	**ECC**
SRR	24	rh	**Service Request Flag** 0_B No service request is pending 1_B A service request is pending
CLRR	25	w	**Request Clear Bit** 0_B No action 1_B Clear SRR; bit value is not stored; read always returns 0; no action if SETR is set in parallel.
SETR	26	w	**Request Set Bit** 0_B No action 1_B Set SRR and SWS; SRR bit value is not stored; read always returns 0; no action if CLRR is set in parallel.
IOV	27	rh	**Interrupt Trigger Overflow Bit** 0_B No Interrupt Trigger Overflow detected 1_B Interrupt Overflow Detected.
IOVCLR	28	w	**Interrupt Trigger Overflow Clear Bit** 0_B No action 1_B Clear IOV; bit value is not stored; read always returns 0.

Field	Bits	Type	Description
SWS	29	rh	**SW Sticky Bit** 0_B No interrupt was initiated via SETR 1_B Interrupt was initiated via SETR
SWSCLR	30	w	**SW Sticky Clear Bit** 0_B No action 1_B Clear SWS; bit value is not stored; read always returns 0.
0	31, [23:21], [15:12], [9:8]	r	**Reserved** Read as 0; should be written with 0.

SRC_GPT120T5
GPT120 T5 Overflow/Underflow Service Request
(0470_H) **Reset Value: $0000\ 0000_H$**

SRC_GPT120T6
GPT120 T6 Overflow/Underflow Service Request
(0474_H) **Reset Value: $0000\ 0000_H$**

SRC_STM0SR0
System Timer 0 Service Request 0
(0490_H) **Reset Value: $0000\ 0000_H$**

SRC_STM0SR1
System Timer 0 Service Request 1
(0494_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**SWSCLR**	**SWS**	**IOVCLR**	**IOV**	**SETR**	**CLRR**	**SRR**	**0**			**ECC**				
r	w	rh	w	rh	w	w	rh	r			rwh				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				**TOS**	**SRE**	**0**		**SRPN**							
r				rw	rw	r		rw							

Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number** 00_H Service request is on lowest priority 01_H Service request is one before lowest priority ... ... FF_H Service request is on highest priority *Note: For a CPU 01H is the lowest priority as 00H is never serviced. For the DMA 00H triggers channel 0*
SRE	10	rw	**Service Request Enable** 0_B Service request is disabled 1_B Service request is enabled
TOS	11	rw	**Type of Service Control** 0_H CPU0 service is initiated 1_H DMA service is initiated
ECC	[20:16]	rwh	**ECC**
SRR	24	rh	**Service Request Flag** 0_B No service request is pending 1_B A service request is pending
CLRR	25	w	**Request Clear Bit** 0_B No action 1_B Clear SRR; bit value is not stored; read always returns 0; no action if SETR is set in parallel.
SETR	26	w	**Request Set Bit** 0_B No action 1_B Set SRR and SWS; SRR bit value is not stored; read always returns 0; no action if CLRR is set in parallel.
IOV	27	rh	**Interrupt Trigger Overflow Bit** 0_B No Interrupt Trigger Overflow detected 1_B Interrupt Overflow Detected.
IOVCLR	28	w	**Interrupt Trigger Overflow Clear Bit** 0_B No action 1_B Clear IOV; bit value is not stored; read always returns 0.

Field	Bits	Type	Description
SWS	29	rh	**SW Sticky Bit** 0_B No interrupt was initiated via SETR 1_B Interrupt was initiated via SETR
SWSCLR	30	w	**SW Sticky Clear Bit** 0_B No action 1_B Clear SWS; bit value is not stored; read always returns 0.
0	31, [23:21], [15:12], [9:8]	r	**Reserved** Read as 0; should be written with 0.

SRC_DMAERR
DMA Error Service Request **($04F0_H$)** **Reset Value: $0000\ 0000_H$**
SRC_DMACHm (m=0-15)
DMA Channel m Service Request
(0500_H+m*4_H) **Reset Value: $0000\ 0000_H$**
SRC_ETH
Ethernet Service Request **($08F0_H$)** **Reset Value: $0000\ 0000_H$**
SRC_CANINTm (m=0-15)
MULTICAN Service Request m **(0900_H+m*4_H)** **Reset Value: $0000\ 0000_H$**
SRC_CAN1INTm (m=0-7)
MULTICAN1 Service Request m **(0940_H+m*4_H)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**SWSCLR**	**SWS**	**IOVCLR**	**IOV**	**SETR**	**CLRR**	**SRR**	**0**			**ECC**				
r	w	rh	w	rh	w	w	rh	r			rwh				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				**TOS**	**SRE**	**0**		**SRPN**							
r				rw	rw	r		rw							

Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number** 00_H Service request is on lowest priority 01_H Service request is one before lowest priority ... ... FF_H Service request is on highest priority *Note: For a CPU 01H is the lowest priority as 00H is never serviced. For the DMA 00H triggers channel 0*
SRE	10	rw	**Service Request Enable** 0_B Service request is disabled 1_B Service request is enabled
TOS	11	rw	**Type of Service Control** 0_H CPU0 service is initiated 1_H DMA service is initiated
ECC	[20:16]	rwh	**ECC**
SRR	24	rh	**Service Request Flag** 0_B No service request is pending 1_B A service request is pending
CLRR	25	w	**Request Clear Bit** 0_B No action 1_B Clear SRR; bit value is not stored; read always returns 0; no action if SETR is set in parallel.
SETR	26	w	**Request Set Bit** 0_B No action 1_B Set SRR and SWS; SRR bit value is not stored; read always returns 0; no action if CLRR is set in parallel.
IOV	27	rh	**Interrupt Trigger Overflow Bit** 0_B No Interrupt Trigger Overflow detected 1_B Interrupt Overflow Detected.
IOVCLR	28	w	**Interrupt Trigger Overflow Clear Bit** 0_B No action 1_B Clear IOV; bit value is not stored; read always returns 0.

Field	Bits	Type	Description
SWS	29	rh	**SW Sticky Bit** 0_B No interrupt was initiated via SETR 1_B Interrupt was initiated via SETR
SWSCLR	30	w	**SW Sticky Clear Bit** 0_B No action 1_B Clear SWS; bit value is not stored; read always returns 0.
0	31, [23:21], [15:12], [9:8]	r	**Reserved** Read as 0; should be written with 0.

SRC_VADCG0SRm (m=0-3)
VADC Group 0 Service Request m
(0980_H+m*4_H) **Reset Value: 0000 0000_H**

SRC_VADCG1SRm (m=0-3)
VADC Group 1 Service Request m
(0990_H+m*4_H) **Reset Value: 0000 0000_H**

SRC_VADCG2SRm (m=0-3)
VADC Group 2 Service Request m
($09A0_H$+m*4_H) **Reset Value: 0000 0000_H**

SRC_VADCG3SRm (m=0-3)
VADC Group 3 Service Request m
($09B0_H$+m*4_H) **Reset Value: 0000 0000_H**

SRC_VADCCG0SRx (x=0-3)
VADC Common Group 0 Service Request x
($0AA0_H$+x*4_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**SWSCLR**	**SWS**	**IOVCLR**	**IOV**	**SETR**	**CLRR**	**SRR**	**0**			**ECC**				
r	w	rh	w	rh	w	w	rh	r			rwh				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				**TOS**	**SRE**	**0**		**SRPN**							
r				rw	rw	r		rw							

Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number** 00_H Service request is on lowest priority 01_H Service request is one before lowest priority ... ... FF_H Service request is on highest priority *Note: For a CPU 01H is the lowest priority as 00H is never serviced. For the DMA 00H triggers channel 0*
SRE	10	rw	**Service Request Enable** 0_B Service request is disabled 1_B Service request is enabled
TOS	11	rw	**Type of Service Control** 0_H CPU0 service is initiated 1_H DMA service is initiated
ECC	[20:16]	rwh	**ECC**
SRR	24	rh	**Service Request Flag** 0_B No service request is pending 1_B A service request is pending
CLRR	25	w	**Request Clear Bit** 0_B No action 1_B Clear SRR; bit value is not stored; read always returns 0; no action if SETR is set in parallel.
SETR	26	w	**Request Set Bit** 0_B No action 1_B Set SRR and SWS; SRR bit value is not stored; read always returns 0; no action if CLRR is set in parallel.
IOV	27	rh	**Interrupt Trigger Overflow Bit** 0_B No Interrupt Trigger Overflow detected 1_B Interrupt Overflow Detected.
IOVCLR	28	w	**Interrupt Trigger Overflow Clear Bit** 0_B No action 1_B Clear IOV; bit value is not stored; read always returns 0.

Field	Bits	Type	Description
SWS	29	rh	**SW Sticky Bit** 0_B No interrupt was initiated via SETR 1_B Interrupt was initiated via SETR
SWSCLR	30	w	**SW Sticky Clear Bit** 0_B No action 1_B Clear SWS; bit value is not stored; read always returns 0.
0	31, [23:21], [15:12], [9:8]	r	**Reserved** Read as 0; should be written with 0.

SRC_ERAYINT0
E-RAY Service Request 0 **($0BE0_H$)** **Reset Value: $0000\ 0000_H$**
SRC_ERAYINT1
E-RAY Service Request 1 **($0BE4_H$)** **Reset Value: $0000\ 0000_H$**
SRC_ERAYTINT0
E-RAY Timer Interrupt 0 Service Request
($0BE8_H$) **Reset Value: $0000\ 0000_H$**
SRC_ERAYTINT1
E-RAY Timer Interrupt 1 Service Request
($0BEC_H$) **Reset Value: $0000\ 0000_H$**
SRC_ERAYNDAT0
E-RAY New Data 0 Service Request **($0BF0_H$)** **Reset Value: $0000\ 0000_H$**
SRC_ERAYNDAT1
E-RAY New Data 1 Service Request **($0BF4_H$)** **Reset Value: $0000\ 0000_H$**
SRC_ERAYMBSC0
E-RAY Message Buffer Status Changed 0 Service Request
($0BF8_H$) **Reset Value: $0000\ 0000_H$**
SRC_ERAYMBSC1
E-RAY Message Buffer Status Changed 1 Service Request
($0BFC_H$) **Reset Value: $0000\ 0000_H$**
SRC_ERAYOBUSY
E-RAY Output Buffer Busy Service Request
($0C00_H$) **Reset Value: $0000\ 0000_H$**
SRC_ERAYIBUSY
E-RAY Input Buffer Busy Service Request
($0C04_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	SWS CLR	SWS	IOVC LR	IOV	SET R	CLR R	SRR	0			ECC				
r	w	rh	w	rh	w	w	rh	r			rwh				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				TOS	SRE	0		SRPN							
r				rw	rw	r		rw							

Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number** 00_H Service request is on lowest priority 01_H Service request is one before lowest priority ... ... FF_H Service request is on highest priority *Note: For a CPU 01H is the lowest priority as 00H is never serviced. For the DMA 00H triggers channel 0*
SRE	10	rw	**Service Request Enable** 0_B Service request is disabled 1_B Service request is enabled
TOS	11	rw	**Type of Service Control** 0_H CPU0 service is initiated 1_H DMA service is initiated
ECC	[20:16]	rwh	**ECC**
SRR	24	rh	**Service Request Flag** 0_B No service request is pending 1_B A service request is pending
CLRR	25	w	**Request Clear Bit** 0_B No action 1_B Clear SRR; bit value is not stored; read always returns 0; no action if SETR is set in parallel.
SETR	26	w	**Request Set Bit** 0_B No action 1_B Set SRR and SWS; SRR bit value is not stored; read always returns 0; no action if CLRR is set in parallel.
IOV	27	rh	**Interrupt Trigger Overflow Bit** 0_B No Interrupt Trigger Overflow detected 1_B Interrupt Overflow Detected.
IOVCLR	28	w	**Interrupt Trigger Overflow Clear Bit** 0_B No action 1_B Clear IOV; bit value is not stored; read always returns 0.

Field	Bits	Type	Description
SWS	29	rh	**SW Sticky Bit** 0_B No interrupt was initiated via SETR 1_B Interrupt was initiated via SETR
SWSCLR	30	w	**SW Sticky Clear Bit** 0_B No action 1_B Clear SWS; bit value is not stored; read always returns 0.
0	31, [23:21], [15:12], [9:8]	r	**Reserved** Read as 0; should be written with 0.

SRC_PMU0m (m=0-1)
PMU 0 Service Request m **($0C30_H$+m*04_H)** **Reset Value: $0000\ 0000_H$**
SRC_HSMm (m=0-1)
HSM Service Request m **($0CC0_H$+m*4_H)** **Reset Value: $0000\ 0000_H$**
SRC_SCUDTS
SCU DTS Busy Service Request **($0CD0_H$)** **Reset Value: $0000\ 0000_H$**
SRC_SCUERUm (m=0-3)
SCU ERU Service Request m **($0CD4_H$+m*4_H)** **Reset Value: $0000\ 0000_H$**
SRC_SMUm (m=0-2)
SMU Service Request m **($0D10_H$+m*4_H)** **Reset Value: $0000\ 0000_H$**
SRC_LMU
LMU Service Request **($0DE0_H$)** **Reset Value: $0000\ 0000_H$**
SRC_EVRWUT
EVR Wake Up Timer Service Request **($0FB0_H$)** **Reset Value: $0000\ 0000_H$**
SRC_EVRSCDC
EVR Supply Service Request **($0FB4_H$)** **Reset Value: $0000\ 0000_H$**
SRC_FFTDONE
FFT Done Service Request **($0FC0_H$)** **Reset Value: $0000\ 0000_H$**
SRC_FFTERR
FFT Error Service Request **($0FC4_H$)** **Reset Value: $0000\ 0000_H$**
SRC_FFTRFS
FFT Ready For Start Service Request **($0FC8_H$)** **Reset Value: $0000\ 0000_H$**
SRC_GPSR0m (m=0-3)
General Purpose Service Request 0m
(1000_H+m*4_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	SWSCLR	SWS	IOVCLR	IOV	SETR	CLRR	SRR	0			ECC				
r	w	rh	w	rh	w	w	rh	r			rwh				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				TOS	SRE	0		SRPN							
r				rw	rw	r		rw							

Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number** 00_H Service request is on lowest priority 01_H Service request is one before lowest priority ... ... FF_H Service request is on highest priority *Note: For a CPU 01H is the lowest priority as 00H is never serviced. For the DMA 00H triggers channel 0*
SRE	10	rw	**Service Request Enable** 0_B Service request is disabled 1_B Service request is enabled
TOS	11	rw	**Type of Service Control** 0_H CPU0 service is initiated 1_H DMA service is initiated
ECC	[20:16]	rwh	**ECC**
SRR	24	rh	**Service Request Flag** 0_B No service request is pending 1_B A service request is pending
CLRR	25	w	**Request Clear Bit** 0_B No action 1_B Clear SRR; bit value is not stored; read always returns 0; no action if SETR is set in parallel.
SETR	26	w	**Request Set Bit** 0_B No action 1_B Set SRR and SWS; SRR bit value is not stored; read always returns 0; no action if CLRR is set in parallel.
IOV	27	rh	**Interrupt Trigger Overflow Bit** 0_B No Interrupt Trigger Overflow detected 1_B Interrupt Overflow Detected.
IOVCLR	28	w	**Interrupt Trigger Overflow Clear Bit** 0_B No action 1_B Clear IOV; bit value is not stored; read always returns 0.

Field	Bits	Type	Description
SWS	29	rh	**SW Sticky Bit** 0_B No interrupt was initiated via SETR 1_B Interrupt was initiated via SETR
SWSCLR	30	w	**SW Sticky Clear Bit** 0_B No action 1_B Clear SWS; bit value is not stored; read always returns 0.
0	31, [23:21], [15:12], [9:8]	r	**Reserved** Read as 0; should be written with 0.

SRC_GTMAEIIRQ
GTM AEI Shared Service Request (1600_H) Reset Value: 0000 0000$_H$
SRC_GTMERR
GTM Error Service Request (1770_H) Reset Value: 0000 0000$_H$
SRC_GTMTIM0m (m=0-7)
GTM TIM0 Shared Service Request m
(1780_H+m*4_H) Reset Value: 0000 0000$_H$
SRC_GTMTOM0m (m=0-7)
GTM TOM0 Shared Service Request m
($1B80_H+m*4_H$) Reset Value: 0000 0000$_H$
SRC_GTMTOM1m (m=0-7)
GTM TOM1 Shared Service Request m
($1BA0_H+m*4_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**SWS CLR**	**SWS**	**IOVC LR**	**IOV**	**SET R**	**CLR R**	**SRR**	**0**			**ECC**				
r	w	rh	w	rh	w	w	rh	r			rwh				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				**TOS**	**SRE**	**0**		**SRPN**							
r				rw	rw	r		rw							

Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number** 00_H Service request is on lowest priority 01_H Service request is one before lowest priority ... ... FF_H Service request is on highest priority *Note: For a CPU 01H is the lowest priority as 00H is never serviced. For the DMA 00H triggers channel 0*
SRE	10	rw	**Service Request Enable** 0_B Service request is disabled 1_B Service request is enabled
TOS	11	rw	**Type of Service Control** 0_H CPU0 service is initiated 1_H DMA service is initiated
ECC	[20:16]	rwh	**ECC**
SRR	24	rh	**Service Request Flag** 0_B No service request is pending 1_B A service request is pending
CLRR	25	w	**Request Clear Bit** 0_B No action 1_B Clear SRR; bit value is not stored; read always returns 0; no action if SETR is set in parallel.
SETR	26	w	**Request Set Bit** 0_B No action 1_B Set SRR and SWS; SRR bit value is not stored; read always returns 0; no action if CLRR is set in parallel.
IOV	27	rh	**Interrupt Trigger Overflow Bit** 0_B No Interrupt Trigger Overflow detected 1_B Interrupt Overflow Detected.
IOVCLR	28	w	**Interrupt Trigger Overflow Clear Bit** 0_B No action 1_B Clear IOV; bit value is not stored; read always returns 0.

Field	Bits	Type	Description
SWS	29	rh	**SW Sticky Bit** 0_B No interrupt was initiated via SETR 1_B Interrupt was initiated via SETR
SWSCLR	30	w	**SW Sticky Clear Bit** 0_B No action 1_B Clear SWS; bit value is not stored; read always returns 0.
0	31, [23:21], [15:12], [9:8]	r	**Reserved** Read as 0; should be written with 0.

17 System Timer (STM)

This chapter describes the System Timer (STM). The TC21x/TC22x/TC23x's STM is designed for global system timing applications requiring both high precision and long period.

17.1 Overview

The STM has the following features:

- Free-running 64-bit counter
- All 64 bits can be read synchronously
- Different 32-bit portions of the 64-bit counter can be read synchronously
- Flexible service request generation based on compare match with partial STM content
- Counting starts automatically after an Application Reset
- STM registers are reset by an Application Reset if bit ARSTDIS.STMxDIS is cleared. If bit ARSTDIS.STMxDIS is set, the STM registers are not reset.

Special STM register semantics provide synchronous views of the entire 64-bit counter, or 32-bit subsets at different levels of resolution.

17.2 Operation

The STM is an upward counter, running at frequency f_{STM}. In case of an Application Reset, the STM is reset if bit SCU_ARSTDIS.STMxDIS is cleared. After reset, the STM is enabled and immediately starts counting up. It is not possible to affect the content of the timer during normal operation. The timer registers can only be read but not written to.

The STM can be optionally disabled for power-saving purposes, or suspended for debugging purposes. In suspend mode, the STM clock is stopped but all registers are still readable.

Due to the 64-bit width of the STM, it is not possible to read its entire content with one instruction. It needs to be read with two load instructions. Since the timer would continue to count between the two load operations, there is a chance that the two values read are not consistent (due to possible overflow from the low part of the timer to the high part between the two read operations). To enable a synchronous and consistent reading of the STM content, a capture register (CAP) is implemented. It latches the content of the high part of the STM each time when one of the registers TIM0 to TIM5 is read. Thus, CAP holds the upper value of the timer at exactly the same time when the lower part is read. The second read operation would then read the content of the CAP to get the complete timer value.

The STM can also be read in sections from seven registers, TIM0 through TIM6, that select increasingly higher-order 32-bit ranges of the STM. These can be viewed as individual 32-bit timers, each with a different resolution and timing range.

The content of the 64-bit System Timer can be compared against the content of two compare values stored in the CMP0 and CMP1 registers. Service requests can be generated on a compare match of the STM with the CMP0 or CMP1 registers.

Figure 17-1 provides an overview on the STM module. It shows the options for reading parts of the STM content.

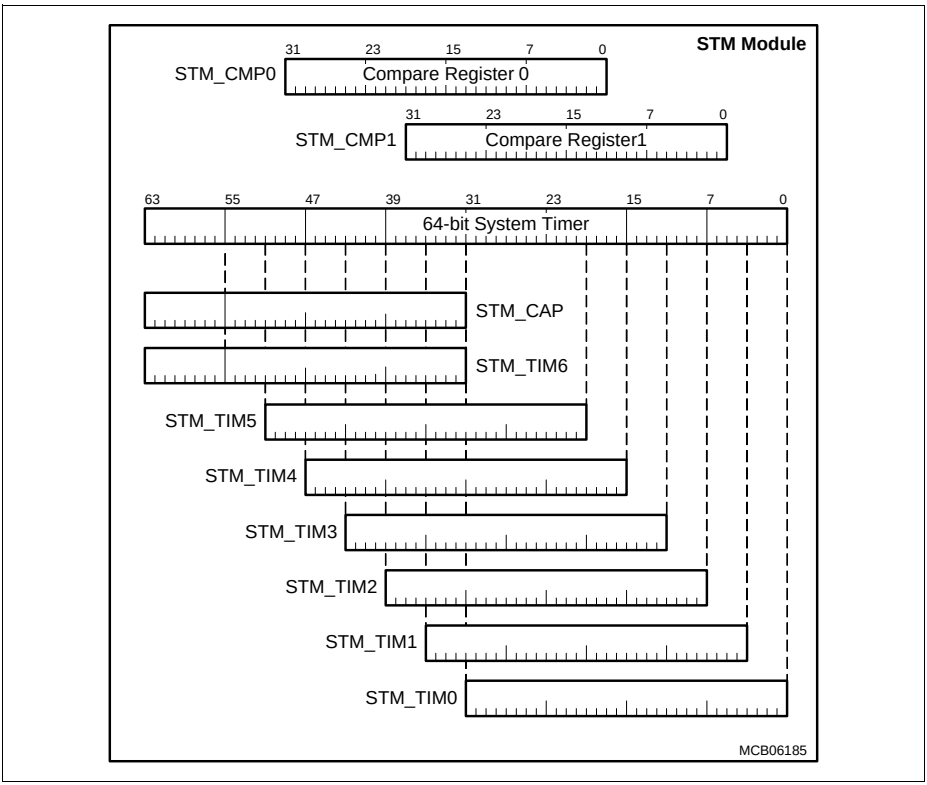


Figure 17-1 General Block Diagram of the STM Module

17.2.1 Compare Register Operation

The content of the 64-bit STM can be compared against the content of two compare values stored in the CMP0 and CMP1 registers. Service requests can be generated on a compare match of the STM with the CMP0 or CMP1 registers.

Two parameters are programmable for the compare operation:

1. The width of the relevant bits in registers CMP0/CMP1 (compare width MSIZEx) that is taken for the compare operation can be programmed from 0 to 31.
2. The first bit location in the 64-bit STM that is taken for the compare operation can be programmed from 0 to 31.

These programming capabilities make compare functionality very flexible. It even makes it possible to detect bit transitions of a single bit n (n = 0 to 31) within the 64-bit STM by setting MSIZE = 0 and MSTART = n.

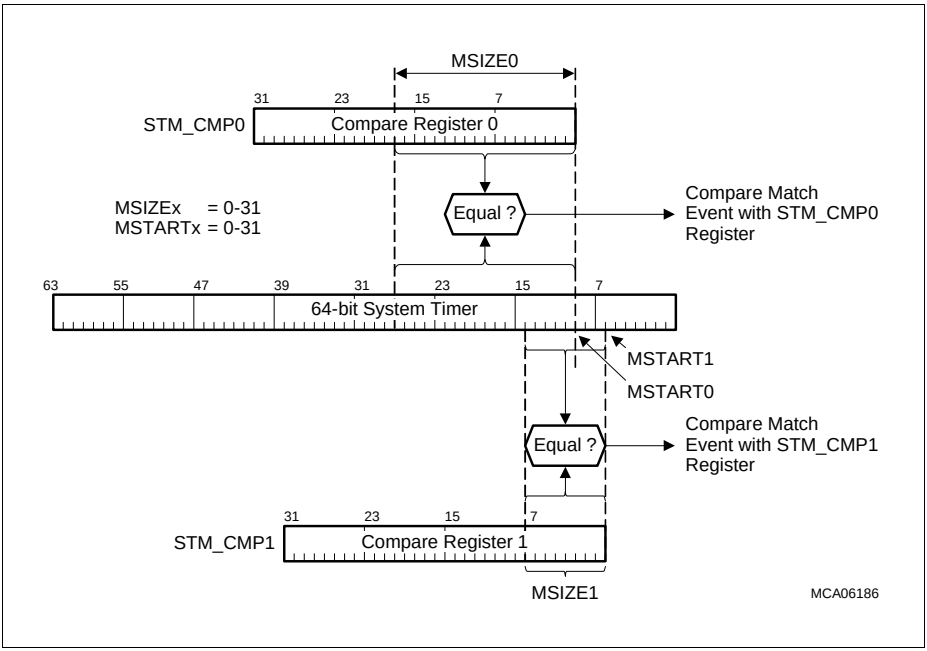


Figure 17-2 Compare Mode Operation

Figure 17-2 shows an example of the compare operation. In this example the following parameters are programmed:

- MSIZE0 = 10001_B = 17_D; MSTART0 = 01010_B = 10_D
- MSIZE1 = 00111_B = 7_D; MSTART1 = 00111_B = 7_D

A compare operation with MSIZE not equal 11111_B always implies that the compared value as stored in the CMP register is right-extended with zeros. This means that in the example of **Figure 17-2**, the compare register content CMP0[17:0] plus ten zero bits right-extended is compared with STM[27:0] with STM[9:0] = 000_H. In case of register CMP1, STM[14:0] with STM[6:0] = 00_H are compared with CMP1[9:0] plus seven zero bits right-extended.

17.2.2 Compare Match Interrupt Control

The compare match interrupt control logic is shown in **Figure 17-3**. Each CMPx register has its compare match interrupt request flag (ICR.CMPxIR) that is set by hardware on a compare match event. The interrupt request flags can be set (ISSR.CMPxIRS) or cleared (ISSR.CMPxIRR) by software. Note that setting ICR.CMPxIR by writing a 1 into ISSR.CMPxIRS does not generate an interrupt at STMIRx. The compare match interrupts from CMP0 and CMP1 can be further directed by ICR.CMPxOS to either output signal STMIR0 or STMIR1.

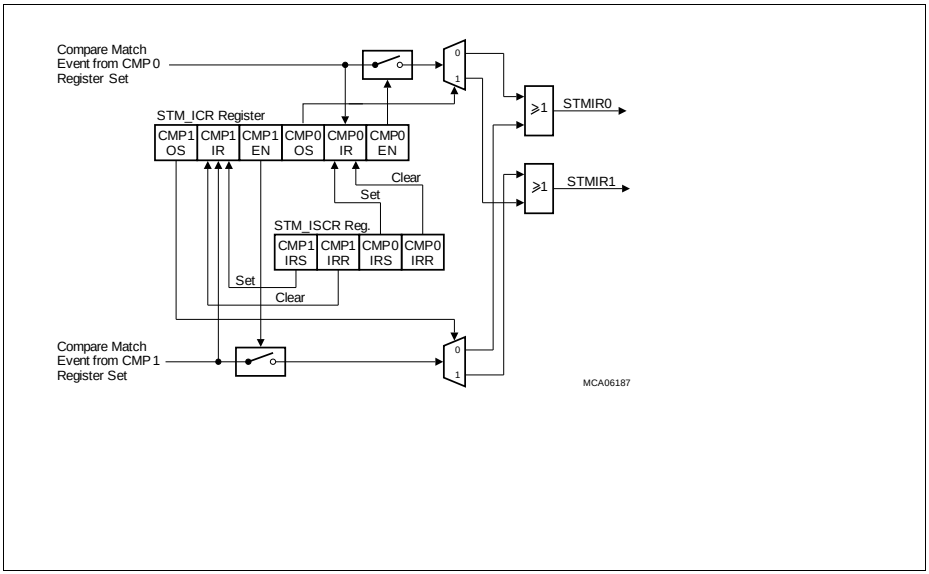


Figure 17-3 STM Interrupt Control

The compare match interrupt flags ICR.CMPxIR are immediately set after an STM reset operation, caused by a compare match event with the reset values of the STM and the compare registers CMPx. This setting of the CMPxIR flags does not directly generate compare match interrupts because the compare match interrupts are automatically disabled after a STM reset operation (CMPxEN = 0). Therefore, before enabling a

compare match interrupt after a STM Application Reset, the CMPxIR flags should be cleared by software (writing register ISSR with CMPxIRR set). Otherwise, undesired compare match interrupt events are triggered.

17.2.3 STM as Reset Trigger

A compare match triggered by an CMP0 event can generate a reset in the system. The reset has to be enable for each STM module individually in register SCU_RSTCON.

17.3 STM Registers

This section describes the STM registers of the STM.

Table 17-1 Registers Address Space

Module	Base Address	End Address	Note
STM0	F000 0000$_H$	F000 00FF$_H$	STM for CPU0

Table 17-2 Registers Overview - DMA Control Registers

Short Name	Description	Offset Addr.	Access Mode		Reset	Description See
			Read	Write		
CLC	Clock Control Register	00$_H$	U, SV	SV, E, P	Application [1)]	Page 17-7
-		04$_H$	BE	BE	-	-
ID	Identification Register	08$_H$	U, SV	BE	Application	Page 17-8
-		0C$_H$	BE	BE	-	-
TIM0	Timer 0 Register	10$_H$	U, SV	BE	Application	Page 17-9
TIM1	Timer 1 Register	14$_H$	U, SV	BE	Application	Page 17-9
TIM2	Timer 2 Register	18$_H$	U, SV	BE	Application	Page 17-10
TIM3	Timer 3 Register	1C$_H$	U, SV	BE	Application	Page 17-10
TIM4	Timer 4 Register	20$_H$	U, SV	BE	Application	Page 17-10
TIM5	Timer 5 Register	24$_H$	U, SV	BE	Application	Page 17-11
TIM6	Timer 6 Register	28$_H$	U, SV	BE	Application	Page 17-11
CAP	Timer Capture Register	2C$_H$	U, SV	BE	Application	Page 17-12
CMP0	Compare Register 0	30$_H$	U, SV	U, SV	Application	Page 17-12
CMP1	Compare Register 1	34$_H$	U, SV	U, SV	Application	Page 17-12

Table 17-2 Registers Overview - DMA Control Registers

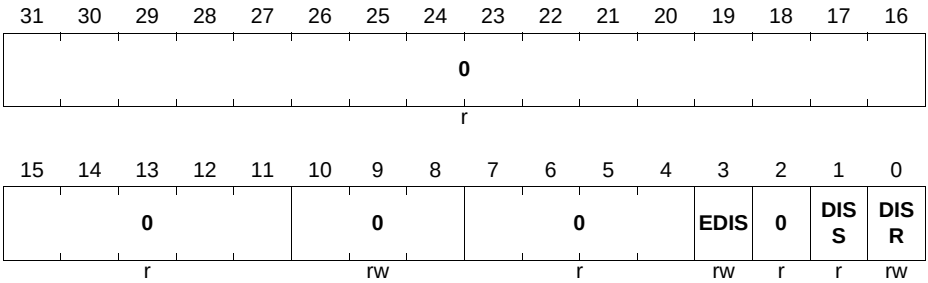
Short Name	Description	Offset Addr.	Access Mode		Reset	Description See
			Read	Write		
CMCON	Compare Match Control Register	38_H	U, SV	U, SV	Application	Page 17-14
ICR	Interrupt Control Register	$3C_H$	U, SV	U, SV	Application	Page 17-16
ISCR	Interrupt Set/Clear Register	40_H	U, SV	U, SV	Application	Page 17-18
-		44_H-$4C_H$	BE	BE	-	-
TIM0SV	Timer 0 Register Second View	50_H	U, SV	BE	Application	Page 17-9
CAPSV	Timer Capture Register Second View	54_H	U, SV	BE	Application	Page 17-12
-		58_H-$E4_H$	BE	BE	-	-
OCS	OCDS Control and Status Register	$E8_H$	U, SV	SV, P	Debug	Page 17-19
KRSTCLR	Reset Status Clear Register	EC_H	U, SV	SV, E, P	Application	Page 17-23
KRST1	Reset Control Register 1	$F0_H$	U, SV	SV, E, P	Application	Page 17-22
KRST0	Reset Control Register 0	$F4_H$	U, SV	SV, E, P	Application	Page 17-21
ACCEN1	Access Enable Register 1	$F8_H$	U, SV	SV, SE	Application	Page 17-20
ACCEN0	Access Enable Register 0	FC_H	U, SV	SV, SE	Application	Page 17-20

1) These registers are reset by an application reset if bit ARSTDIS.STMxDIS is cleared.

17.3.1 Clock Control Register

The STM clock control register is used to switch the STM on or off and to control its input clock rate. After reset, the STM is always enabled and starts counting. The STM can be disabled by setting bit DISR to 1.

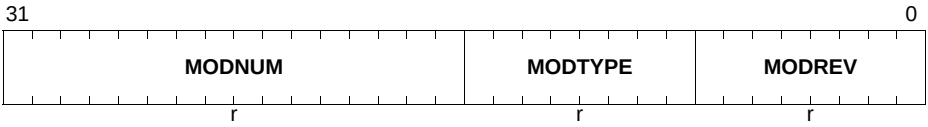
CLC
Clock Control Register **(00_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the STM module. 0_B No disable requested 1_B Disable requested *Note: f_{STM} is generated by the CCU.*
DISS	1	r	**Module Disable Status Bit** Bit indicates the current status of the STM module. 0_B STM module is enabled 1_B STM module is disabled
EDIS	3	rw	**Sleep Mode Enable Control** Used for module sleep mode control.
0	[10:8]	rw	**Reserved** Should be written with 0.
0	2, [7:4], [31:11]	r	**Reserved** Read as 0; should be written with 0.

The STM Module Identification Register ID contains read-only information about the module version.

ID
Module Identification Register **(08_H)** **Reset Value: 0000 C0XX$_H$**



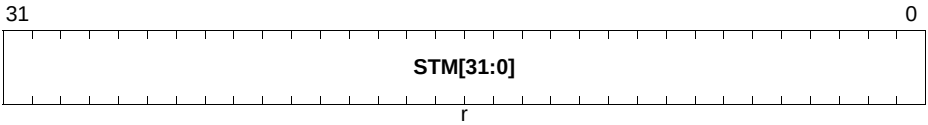
Field	Bits	Type	Description
MODREV	[7:0]	r	**Module Revision Number** MODREV defines the module revision number. The value of a module revision starts with 01_H (first revision).
MODTYPE	[15:8]	r	**Module Type** This bit field defines the module as a 32-bit module: $C0_H$
MODNUM	[31:16]	r	**Module Number Value** This bit field defines the module identification number for the STM: 0000_H

17.3.2 Timer/Capture Registers

Registers TIM0 to TIM6 provide 32-bit views at varying resolutions of the underlying STM counter.

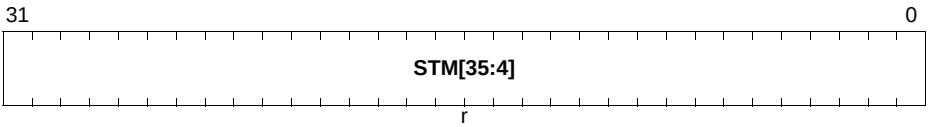
Register TIM0SV address the STM[31:0] bits at a second optional address as TIM0.

TIM0
Timer Register 0 **(10_H)** **Reset Value: $0000\ 0000_H$**
TIM0SV
Timer Register 0 Second View **(50_H)** **Reset Value: $0000\ 0000_H$**



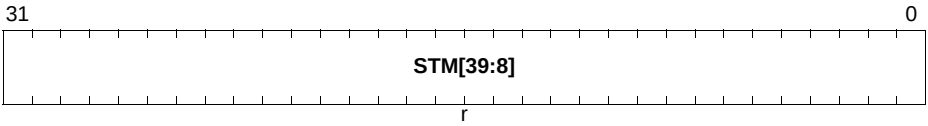
Field	Bits	Type	Description
STM[31:0]	[31:0]	r	**System Timer Bits [31:0]** This bit field contains bits [31:0] of the 64-bit STM.

TIM1
Timer Register 1 **(14_H)** **Reset Value: $0000\ 0000_H$**



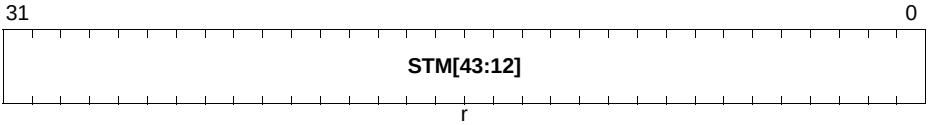
Field	Bits	Type	Description
STM[35:4]	[31:0]	r	**System Timer Bits [35:4]** This bit field contains bits [35:4] of the 64-bit STM.

TIM2
Timer Register 2 **(18_H)** **Reset Value: 0000 0000$_H$**



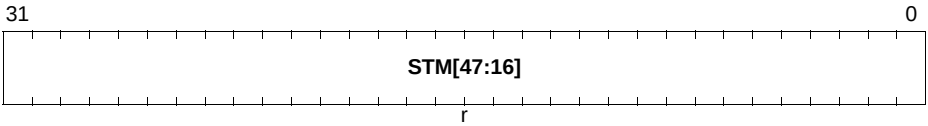
Field	Bits	Type	Description
STM[39:8]	[31:0]	r	**System Timer Bits [39:8]** This bit field contains bits [39:8] of the 64-bit STM.

TIM3
Timer Register 3 **($1C_H$)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
STM[43:12]	[31:0]	r	**System Timer Bits [43:12]** This bit field contains bits [43:12] of the 64-bit STM.

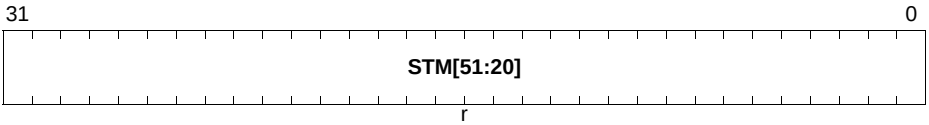
TIM4
Timer Register 4 **(20_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
STM[47:16]	[31:0]	r	**System Timer Bits [47:16]** This bit field contains bits [47:16] of the 64-bit STM.

TIM5

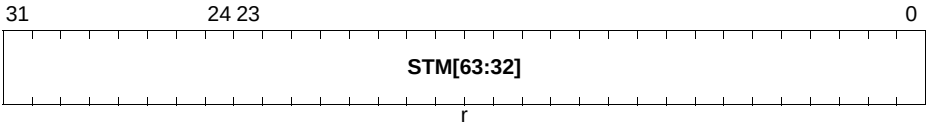
Timer Register 5 **(24_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
STM[51:20]	[31:0]	r	**System Timer Bits [51:20]** This bit field contains bits [51:20] of the 64-bit STM.

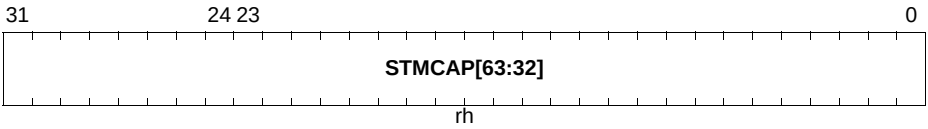
TIM6

Timer Register 6 **(28_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
STM[63:32]	[31:0]	r	**System Timer Bits [63:32]** This bit field contains bits [63:32] of the 64-bit STM.

CAP
Timer Capture Register ($2C_H$) Reset Value: 0000 0000$_H$
CAPSV
Timer Capture Register Second View (54_H) Reset Value: 0000 0000$_H$

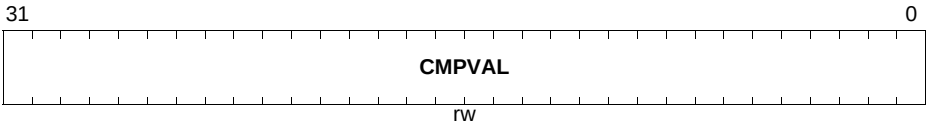


Field	Bits	Type	Description
STMCAP[63:32]	[31:0]	rh	**Captured System Timer Bits [63:32]** The capture register STMCAP always captures the STM bits [63:32] when one of the registers TIM0 to TIM5 and TIM0SV is read. This capture operation is performed in order to enable software to operate with a coherent value of all the 64 STM bits at one time stamp.This bit field contains bits [63:32] of the 64-bit STM. *Note: Reading register TIM0SV capture also the read value for register TIM6. In this way reading TIM0SV followed by CAP6SV delivers the timer values for the first read request.*

17.3.3 Compare Registers

The compare register CMPx holds up to 32-bits; its value is compared to the value of the STM.

CMPx (x = 0-1)
Compare Register x (30_H+x*4_H) Reset Value: 0000 0000$_H$

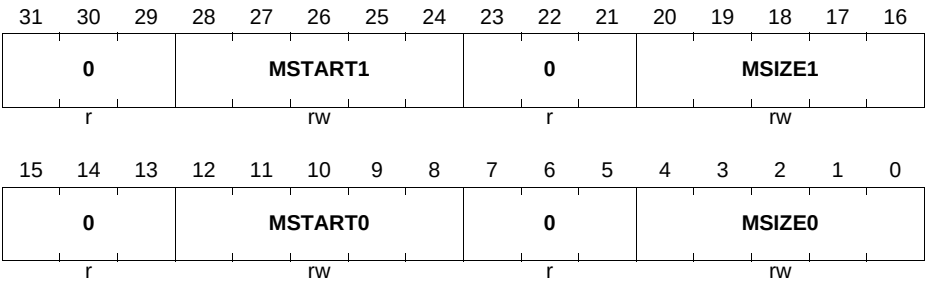


Field	Bits	Type	Description
CMPVAL	[31:0]	rw	**Compare Value of Compare Register x** This bit field holds up to 32 bits of the compare value (right-adjusted).

The STM Compare Match Control Register controls the parameters of the compare logic.

CMCON
Compare Match Control Register

(38_H) **Reset Value: $0000\ 0000_H$**



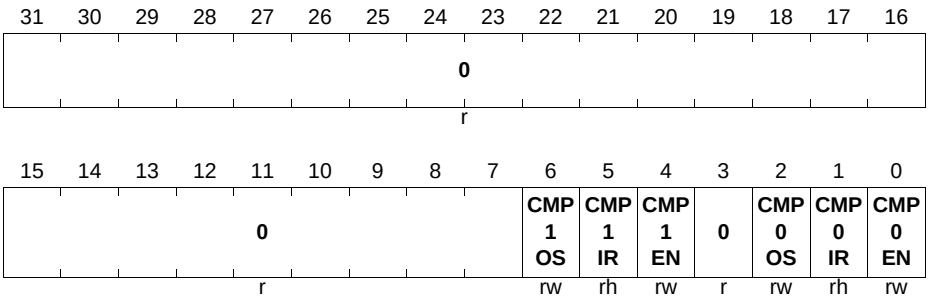
Field	Bits	Type	Description
MSIZE0	[4:0]	rw	**Compare Register Size for CMP0** This bit field determines the number of bits in register CMP0 (starting from bit 0) that are used for the compare operation with the System Timer. 00000_B CMP0[0] used for compare operation 00001_B CMP0[1:0] used for compare operation ... 11110_B CMP0[30:0] used for compare operation 11111_B CMP0[31:0] used for compare operation
MSTART0	[12:8]	rw	**Start Bit Location for CMP0** This bit field determines the lowest bit number of the 64-bit STM that is compared with the content of register CMP0 bit 0. The number of bits to be compared is defined by bit field MSIZE0. 00000_B STM[0] is the lowest bit number 00001_B STM[1] is the lowest bit number ... 11110_B STM[30] is the lowest bit number 11111_B STM[31] is the lowest bit number

Field	Bits	Type	Description
MSIZE1	[20:16]	rw	**Compare Register Size for CMP1** This bit field determines the number of bits in register CMP1 (starting from bit 0) that are used for the compare operation with the System Timer. 00000_B CMP1[0] used for compare operation 00001_B CMP1[1:0] used for compare operation ... 11110_B CMP1[30:0] used for compare operation 11111_B CMP1[31:0] used for compare operation
MSTART1	[28:24]	rw	**Start Bit Location for CMP1** This bit field determines the lowest bit number of the 64-bit STM that is compared with the content of register CMP1 bit 0. The number of bits to be compared is defined by bit field MSIZE1. 00000_B STM[0] is the lowest bit number 00001_B STM[1] is the lowest bit number ... 11110_B STM[30] is the lowest bit number 11111_B STM[31] is the lowest bit number
0	[7:5], [15:13], [23:21], [31:29]	r	**Reserved** Read as 0; should be written with 0.

17.3.4 Interrupt Registers

The two compare match interrupts of the STM are controlled by the STM Interrupt Control Register.

ICR
Interrupt Control Register **($3C_H$)** **Reset Value: 0000 0000$_H$**

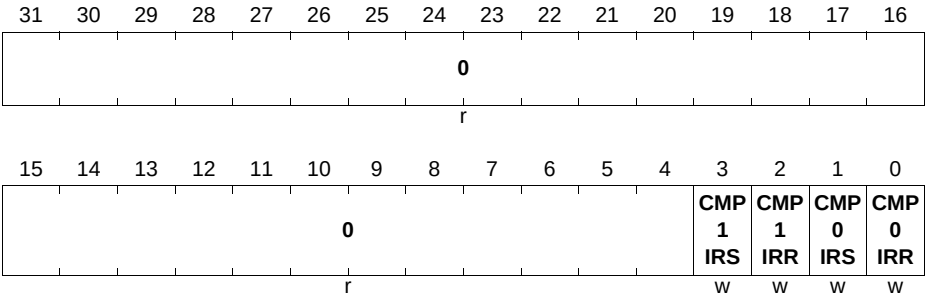


Field	Bits	Type	Description
CMP0EN	0	rw	**Compare Register CMP0 Interrupt Enable Control** This bit enables the compare match interrupt with compare register CMP0. 0_B Interrupt on compare match with CMP0 disabled 1_B Interrupt on compare match with CMP0 enabled
CMP0IR	1	rh	**Compare Register CMP0 Interrupt Request Flag** This bit indicates whether or not a compare match interrupt request of compare register CMP0 is pending. CMP0IR must be cleared by software. 0_B A compare match interrupt has not been detected since the bit has been cleared for the last time. 1_B A compare match interrupt has been detected. CMPIR0 must be cleared by software and can be set by software, too (see CMPISCR register). After a STM reset operation, CMP0IR is immediately set as a result of a compare match event with the reset values of the STM and the compare registers CMP0.

Field	Bits	Type	Description
CMP0OS	2	rw	**Compare Register CMP0 Interrupt Output Selection** This bit determines the interrupt output that is activated on a compare match event of compare register CMP0. 0_B Interrupt output STMIR0 selected 1_B Interrupt output STMIR1 selected
CMP1EN	4	rw	**Compare Register CMP1 Interrupt Enable Control** This bit enables the compare match interrupt with compare register CMP1. 0_B Interrupt on compare match with CMP1 disabled 1_B Interrupt on compare match with CMP1 enabled
CMP1IR	5	rh	**Compare Register CMP1 Interrupt Request Flag** This bit indicates whether or not a compare match interrupt request of compare register CMP1 is pending. CMP1IR must be cleared by software. 0_B A compare match interrupt has not been detected since the bit has been cleared for the last time. 1_B A compare match interrupt has been detected. CMPIR1 must be cleared by software and can be set by software, too (see CMPISCR register). After a STM reset, CMP1IR is immediately set as a result of a compare match event with the reset values of the STM and the compare register CMP1.
CMP1OS	6	rw	**Compare Register CMP1 Interrupt Output Selection** This bit determines the interrupt output that is activated on a compare match event of compare register CMP1. 0_B Interrupt output STMIR0 selected 1_B Interrupt output STMIR1 selected
0	3, [31:7]	r	**Reserved** Read as 0; should be written with 0.

The bits in the STM Interrupt Set/Clear Register make it possible to set or cleared the compare match interrupt request status flags of register ICR.

ISCR
Interrupt Set/Clear Register **(40_H)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CMP0IRR	0	w	**Reset Compare Register CMP0 Interrupt Flag** 0_B Bit ICR.CMP0IR is not changed. 1_B Bit ICR.CMP0IR is cleared.
CMP0IRS	1	w	**Set Compare Register CMP0 Interrupt Flag** 0_B Bit ICR.CMP0IR is not changed. 1_B Bit ICR.CMP0IR is set. The state of bit CMP0IRR is “don’t care” in this case.
CMP1IRR	2	w	**Reset Compare Register CMP1 Interrupt Flag** 0_B Bit ICR.CMP1IR is not changed. 1_B Bit ICR.CMP1IR is cleared.
CMP1IRS	3	w	**Set Compare Register CMP1 Interrupt Flag** 0_B Bit ICR.CMP1IR is not changed. 1_B Bit ICR.CMP1IR is set. The state of bit CMP1IRR is “don’t care” in this case.
0	[31:4]	r	**Reserved** Read as 0; should be written with 0.

Note: Reading register ISCR always returns $0000\ 0000_H$.

17.3.5 Interface Registers

OCDS Control and Status Register (OCS)

The OCDS Control and Status (OCS) register is cleared by Debug Reset.

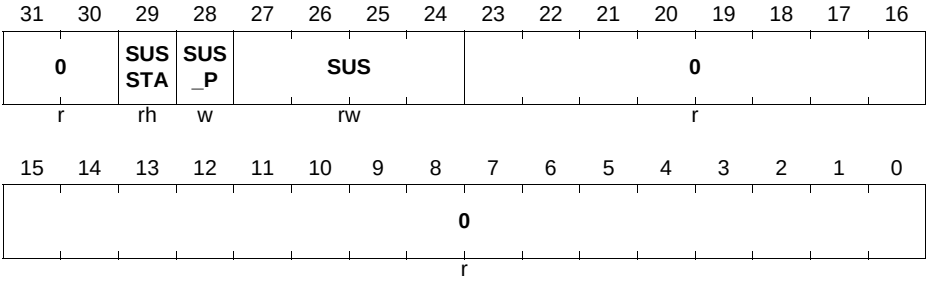
The OCS register can only be written when the OCDS is enabled.

If OCDS is being disabled, the OCS register value will not change.

When OCDS is disabled the OCS suspend control is ineffective.

Write access is 32 bit wide only and requires Supervisor Mode.

OCS
OCDS Control and Status **($E8_H$)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
SUS	[27:24]	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0_H Will not suspend 1_H Reserved, do not use this combination 2_H 64-bit counter will be stopped **others**, Reserved, do not use this combination *Note:* For details see the OCDS chapter.
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	[23:0], [31:30]	r	**Reserved** Read as 0; must be written with 0.

The Access Enable Register 0 controls write access for transactions with the on chip bus master TAG ID 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000_B, EN1 -> TAG ID 000001_B , ... ,EN31 -> TAG ID 011111_B.

ACCEN0

Access Enable Register 0 **(FC_H)** **Reset Value: FFFF FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN 31	**EN 30**	**EN 29**	**EN 28**	**EN 27**	**EN 26**	**EN 25**	**EN 24**	**EN 23**	**EN 22**	**EN 21**	**EN 20**	**EN 19**	**EN 18**	**EN 17**	**EN 16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN 15	**EN 14**	**EN 13**	**EN 12**	**EN 11**	**EN 10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

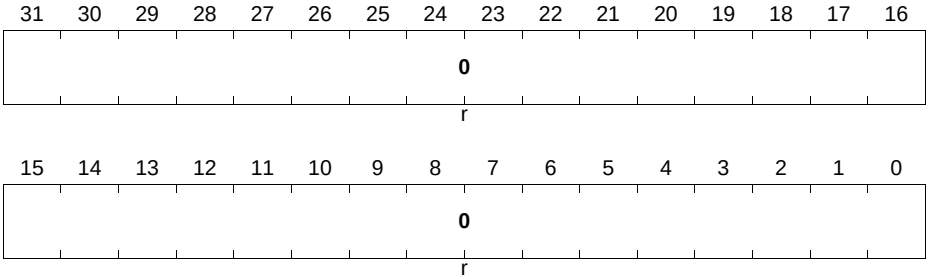
Field	Bits	Type	Description
ENn (n = 0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Access Enable Register (ACCEN1)

The Access Enable Register 1 controls write access for transactions with the on chip bus master TAG ID 100000_B to 111111_B (see On Chip Bus chapter for the products TAG ID master peripheral mapping).

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001_B, ... , EN31 -> TAG ID 111111_B.

ACCEN1
Access Enable Register 1 **($F8_H$)** **Reset Value: 0000 0000$_H$**

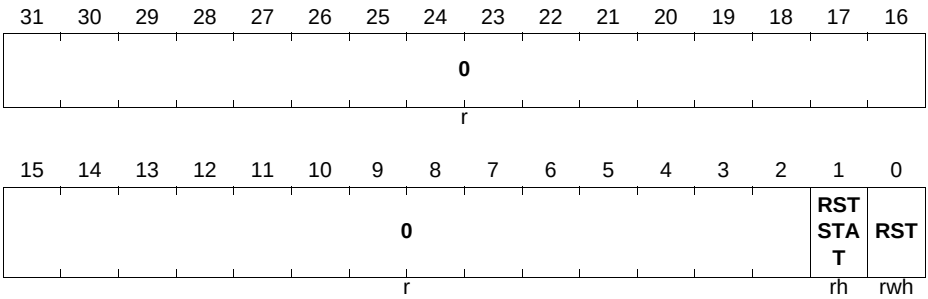


Field	Bits	Type	Description
0	[31:0]	r	**Reserved** Read as 0; should be written with 0.

The Kernel Reset Register 0 is used to reset the related module kernel. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers related to the module kernel that should be reset (kernel 0 or kernel 1). In order support modules with two kernel the BPI_FPI provides two set of kernel reset registers. The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing to it with ´1´.

KRST0
Kernel Reset Register 0 **($F4_H$)** **Reset Value: 0000 0000$_H$**

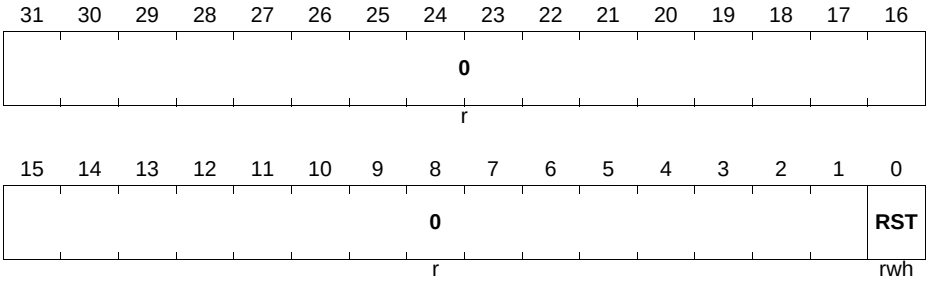


Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed.
RSTSTAT	1	rw	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. 0_B No kernel reset was executed 1_B Kernel reset was executed This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register.
0	[31:2]	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1 (KRST1)

The Kernel Reset Register 1 is used to reset the STM kernel. STM kernel registers related to the Debug Reset (Class 1) are not influenced. To reset the STM kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset registers. The RST bit will be cleared with the end of the BPI kernel reset sequence.

KRST1
Kernel Reset Register 1 **($F0_H$)** **Reset Value: $0000\ 0000_H$**

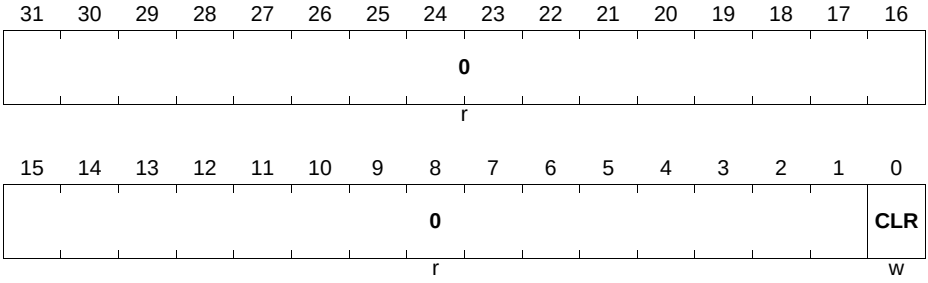


Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) after the kernel reset was executed.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

The Kernel Reset Register Clear register is used to clear the Kernel Reset Status bit (<>_KRST0.RSTSTAT).

KRSTCLR

Kernel Reset Status Clear Register (EC_H) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT Read always as 0.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

18 On-Chip Debug Support (OCDS)

AURIX devices contain powerful resources for debugging and performance optimization. They provide high visibility and controllability of software, hardware and system, especially also under hard real-time constraints. The resources are either embedded in specific modules (e.g. breakpoint logic of CPUs) or part of the central CERBERUS module.

Figure 18-1 shows the AURIX family concept of the OCDS with all potentially usable pins. The same OCDS function is always on the same port pin including full DAP/JTAG functionality but the number of $\overline{\text{TGI}}$/$\overline{\text{TGO}}$ trigger pins depends on the device type.

Trace functionality is available for TC29x with miniMCDS, for all other AURIX devices only with the associated Emulation Device (ED) with MCDS.

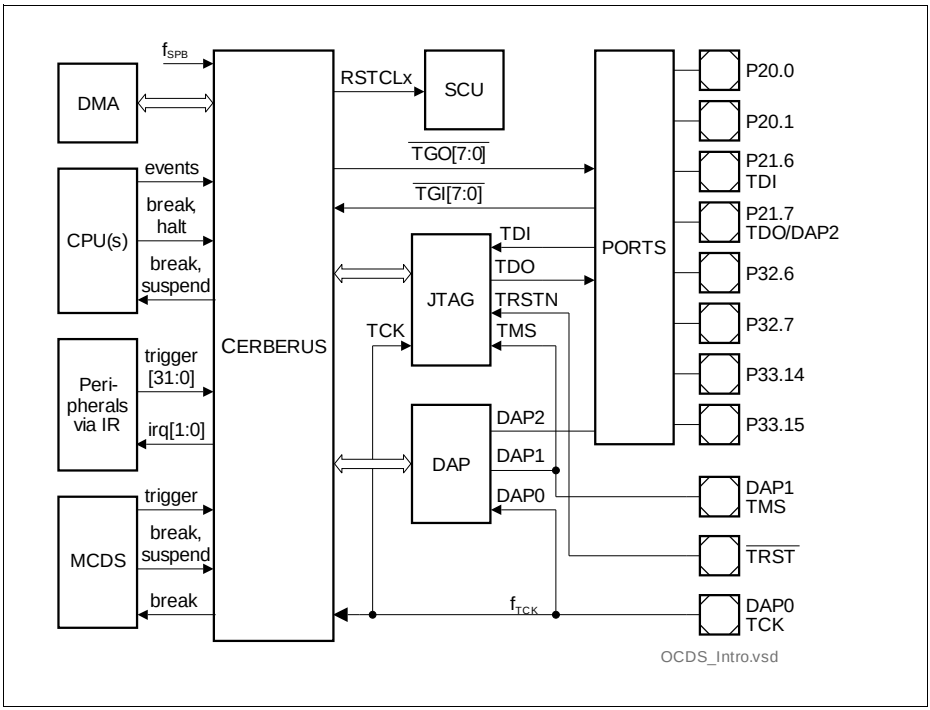


Figure 18-1 OCDS Block Diagram (Family Concept)

When acting as bus master CERBERUS shares the master interfaces of the DMA (using a different master tag) to access memories and SFRs attached to the SPB or SRI via the shortest possible path.

18.1 Feature List

Please also refer to debug specific features in other modules, e.g.

- Eight hardware breakpoints for TriCore, based on instruction or data address.
- Unlimited number of software breakpoints (DEBUG instruction).
- Trigger generated by the access to a specific bus address by any bus master
- Peripheral trigger and trace with OCDS Trigger Bus (OTGB)
  - Peripherals provide vectors (Trigger Sets) of their most interesting signals
  - Signals can be routed to trigger pins
  - Flexible tracing of signal vectors from one or two peripherals in parallel (ED)
- Dedicated interrupt resources to handle debug events, both local inside the CPUs (breakpoint trap, software interrupt) and global (triggered by CERBERUS), e.g. for implementing monitor programs

Central functions implemented by CERBERUS

- Run/stop and single-step execution independently for each CPU.
- Run/stop and time-step execution of the complete device using the Trigger Switch.
- Automatic suspension of the SCU watchdogs when the device is halted by the tool.
- All kinds of reset can be requested using only the tool interface.
- Halt-after-Reset for repeatable debug sessions.
- Tool access to all SFRs and internal memories independent of the CPUs.
- Bus priority of CERBERUS can be chosen dynamically to minimize real-time impact.
- Up to 8 package pins can be used optional as with Trigger In/Out ($\overline{\text{TGI}}$/$\overline{\text{TGO}}$).
- Central OCDS Trigger Switch (OTGS) with 7 independent Trigger Lines to collect debug events from various sources (all CPUs, all interrupt requesters, bus controllers, several complex peripherals, MCDS, trigger input pins) and distribute them selectively to all CPUs and trigger output pins (extension of the former Break Switch concept).
- Central Suspend Switch using up to three Lines of the Trigger Switch infrastructure. This allows to selectively suspend all or only part of the CPUs and peripherals instead of halting them as reaction to any debug event.
- Access to all OCDS resources also for the CPUs themselves for debug tools integrated into the application code.
- Triggered Transfer of data for simple variable tracing.
- A dedicated trigger bank (TRIG) with 96/512 independent status bits is provided to post requests at a central location from application code to the tool.
- The tool is notified automatically when the trigger bank is updated by any processor. No polling via a system bus is required.

Tool Interfaces

Several options exist for the communication channel between tools and TC21x/TC22x/TC23x:

- DAP and JTAG are clocked by the tool.
- Two pin DAP (Device Access Port) protocol for long connections or noisy environments.
- Three unidirectional pin DAP mode for off-chip transceiver integration (e.g. LVDS).
- Three pin DAP Wide Mode for high bandwidth needs.
- DAP bit clock can have any frequency up to 160 MHz.
- 15 MByte/s for block read or write, 30 MByte/s in Wide Mode
- Optimized random memory accesses (read word within 0.5 µs at 160 MHz).
- CAN (plus software linked into the application code) for embedded purposes with lower bandwidth requirements.
- Four pin JTAG (IEEE 1149.1) for standard manufacturing tests.
- Hot attach (i.e. physical disconnect/reconnect of the host connection without reset) for all interfaces.
- Infineon standard DAS (Device Access Server) implementation for seamless, transparent and parallel tool access over any supported interface.
- Lock mechanism to prevent unauthorized tool access to application code.
- DAP over CAN Messages (DXCM) u, TC23x/22x only)

FAR Support

To efficiently locate and identify faults after integration of an AURIX device into a system special functions are available:

- Boundary Scan (IEEE 1149.1) via JTAG or DAP.
- SSCM (Single Scan Chain Mode) for structural scan testing of the chip itself.
- DXCPL (DAP over CAN Physical Layer) via CAN pins (AP32264)

18.2 Family Overview

The OCDS architecture and features are very consistent over the whole AURIX family. This makes it easy for tool partners and users to switch between different devices. **Table 18-1** lists all OCDS and the main Emulation Device differences of the devices, and **Table 18-2** the JTAG IDs. The associated ED step has the same JTAG ID.

Table 18-1 OCDS Differences of AURIX Devices

Feature	TC29x	TC27x	TC26x	TC24x	TC23x	TC22x/1x
TRIG triggers	512	512	512	512	96	96
DAP Over CAN Msg. (DXCM)	-	-	-	-	yes	yes
miniMCDS	yes	-	-	-	-	-
Emulation Device (ED)	9xED	7xED	6xED	(6xED)	3xED	(3xED)
ED packages B=BGA, Q=QFP	B 292 B 416 B 516	B 292 Q 176	Q 176 Q 144	Q 144	Q 144	Q 144
ED's TDI pin replaced by VDDPSB	-	Q 176	Q 176 Q 144	Q 144	-	-

Table 18-2 JTAG IDs of AURIX Devices

	A-Step	B-Step	C-Step	D-Step
TC29x	$101E9083_H$	$201E9083_H$		
TC27x	$201DA083_H$	$301DA083_H$	$401DA083_H$	$501DA083_H$
TC26x	$101E8083_H$	$201E8083_H$		
TC24x	$101EA083_H$			
TC23x	10200083_H			
TC22x	10201083_H			
TC21x	10202083_H			

18.3 Tool Interface Recommendations

AURIX devices are well supported by many tool partners for different types of tools. Standard tool interface for debug, measurement and calibration is DAP due to its reduced pin-count, higher performance (3-6x) and higher robustness (CRC) than JTAG. Please note that the full "JTAG" boundary scan functionality is also available with DAP, it is supported however only by specific tool providers. Please refer to application note AP24003 for more information about the standard DAP connector and board design for high speed DAP. **Table 18-3** lists all pins and considerations for connecting tools.

Table 18-3 Tool Relevant Device Pins of AURIX Family

Pins	Remark
TRST	DAP: Has to be high at PORST pin release. Can be statically connected to VDDP3 (e.g. in the development phase only) with these effects: • Causes a static current due to pull-down element on chip • VDDSB and VDDPSB supplies need to be ensured • DXCPL is disabled JTAG: Needs to be controlled by the tool via the tool connector.
DAP0/TCK	DAP: Please consider AP24003 for high speed DAP
DAP1/TMS	
DAP2/TDO TGI3/TGO3	DAP: Needed for three pin modes like high bandwidth Wide Mode. The standard DAP connector (AP24003) allows to use this pin on demand either for Wide Mode e.g. for measurement or as trigger pin for system debugging.
TDI/(VDDPSB) TGI2/TGO2	For certain EDs in QFP packages (**Table 18-1**) this pin is the VDDPSB supply pin and needs to be supplied with 3.3 V (e.g. VDDP3). Please note for the “TDI is VDDPSB” case: • JTAG is not available for such EDs • VDDP3 connection can be hardwired. This is compatible for PD and ED, however TGI2/TGO2 is then also unavailable for the PD.
VDDSB	1.3 V supply of the ED memory. This supply can be separate of VDD if standby functionality is needed. Please consider the additional VDDSB current for your supply.
VDDPSB	3.3 V supply for the DAP pins of EDs. Can be connected to VDDP3 or supplied by the tool.
TGIx/TGOx	Optional trigger pins, overlaid to port pins. Availability depends on device and package type.
AGBT_xyz	AGBT high-speed serial pins in the center ball matrix of EDs in BGA packages. Please connect to VSS if no AGBT is needed.

Note: Please ensure that VDDSB and VDDPSB are always supplied in regular operation (VDD supplied, TRST/PORST inactive) to avoid internal cross currents which can damage the ED device. For instance connect VDDSB to VDD and VDDPSB to VDDP3.

For more information please contact your Infineon support.

18.4 Debug Access Server (DAS)

The DAS API provides an abstraction of the physical device interface for tool access. The key paradigm of DAS is to read or write data in one or several address spaces of the target device.

DAS Features

- Standard interface for all types of tools
- Efficient and robust methods for data transfer
- Standardized system security support (authorization)
- Several independent tools can share the same physical interface
- Product chip address space is represented with DAS address map 0, EEC with 1
- Infineon's miniWiggler supports DAP, JTAG, SWD and SPD

DAS is not device specific. It can be used for all Infineon 8-, 16- and 32-bit microcontrollers with DAP, JTAG, SWD, or SPD interface. For more information please refer to **www.infineon.com/DAS**.

19 Asynchronous/Synchronous Interface (ASCLIN)

The main purpose of the ASCLIN module is to provide asynchronous serial communication with external devices using only data-in, data-out signals.

The focus of the module is set to fast and flexible communication: either fast point-to-point or master-to-many slaves communication using the LIN protocol.

Additionally, the module supports the synchronous SPI communication.

Figure 19-1 shows an overview of the ASCLIN module.

Note: The RX, TX, RTS, CTS, SCLKO, SLSO signal names are prefixed with "A" in order to achieve unique names on chip level, distinct from signal names of other communication modules.

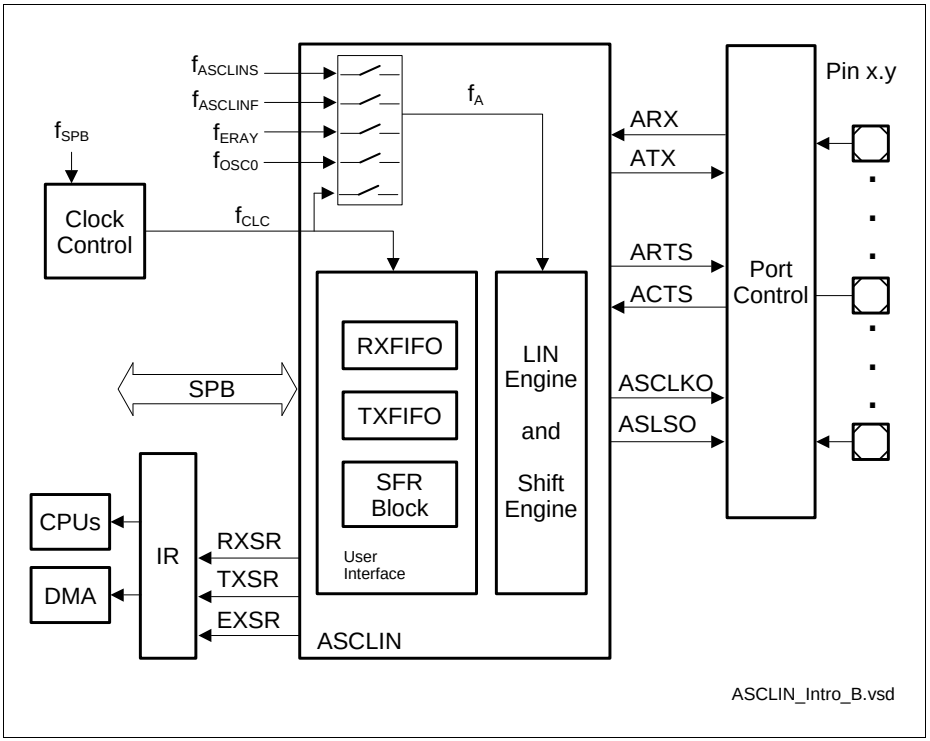


Figure 19-1 Block Diagram of the ASCLIN module.

>> **Registers Overview**

>> **BPI_FPI Module Registers**

19.1 Feature List

This section describes the features of the ASCLIN module.

General Features

- 16 bytes TxFIFO
- 16 bytes RxFIFO
- Pack / unpack capabilities of the Tx and Rx FIFO
- Interrupt generation
  - On a configurable transmit FIFO level
  - On a configurable receive FIFO level
  - On an error condition (frame, parity, overrun error)
  - On various module internal events (end of ASC/SPI frame, LIN events)
- Interrrupt signals capable of triggering either a CPU or a DMA
- Programmable oversampling of 4 to 16 times per bit
- Programmable sampling point position
- Programmable digital glitch filter and median filter for incoming bit stream
- Shift direction LSB first
- Internal loop-back mode

Standard ASC Features

- Full-duplex asynchronous operating modes
  - 7-bit, 8-bit or 9-bit (or up to 16-bit) data frames, LSB first
  - Parity-bit generation/checking
  - One or two stop bits
  - Max baud rate f_A / 16 (6.25 MBaud @ 100 MHz f_A module clock)
  - Min. baud rate f_A / 268 435 456 (0.37 Baud @ 100 MHz f_A module clock)
- Optional RTS / CTS handshaking
- Support of
  - JASO D 903

Extended ASC Features

- Programmable oversampling of 4 to 16 times per bit
  - module capability of up to 4 times higher baud rates (f_A / 4) then the standard ASC
  - system considerations like pad type, incoming signal quality and accumulated PLL jitter can lead to constraints regarding the usable oversampling ratios (for example f_A / 8 = 200MHz / 8 = 25 MBaud)
- Programmable sampling point position

LIN Features

- Support of

  - LIN version 1.3
  - LIN version 2.0
  - LIN version 2.1 and
  - J2602
- Break detection
- Break injection
- Sync field generation
- Auto baud detection based on Sync Field measurement
- Optional Collision detection, required for LIN version 2.1
- LIN Watchdogs
  - Header time-out
  - Frame or Response time-out
- Stuck at zero/one monitoring
- Bus idle time monitoring
- Wake-Up
- Minimum CPU load in master mode
  - Single interrupt indicating the end of the frame
- Minimum CPU load in slave mode
  - Single interrupt at the end of the header reception
  - Single interrupt at the end of the response or end of frame
- Standard operation with one interrupt per transmitted or received byte supported

SPI Features

- SPI master modes (slave mode not supported):
  - Four-wire or three-wire (with / without slave select output signal)
- Up to 16-bit data width
- Full-duplex and half-duplex
  - Min. baud rate f_A / 268 435 456 MBaud (= 0.37 Baud @ 100 MHz f_A module clock)
  - Max. baud rate f_A / 4 MBaud (= 25 MBaud @ 100 MHz f_A module clock)
- Programmable leading and trailing delays

19.2 Overview

This section shows the essential sub blocks of the module when used as standard ASC, as LIN or as SPI master.

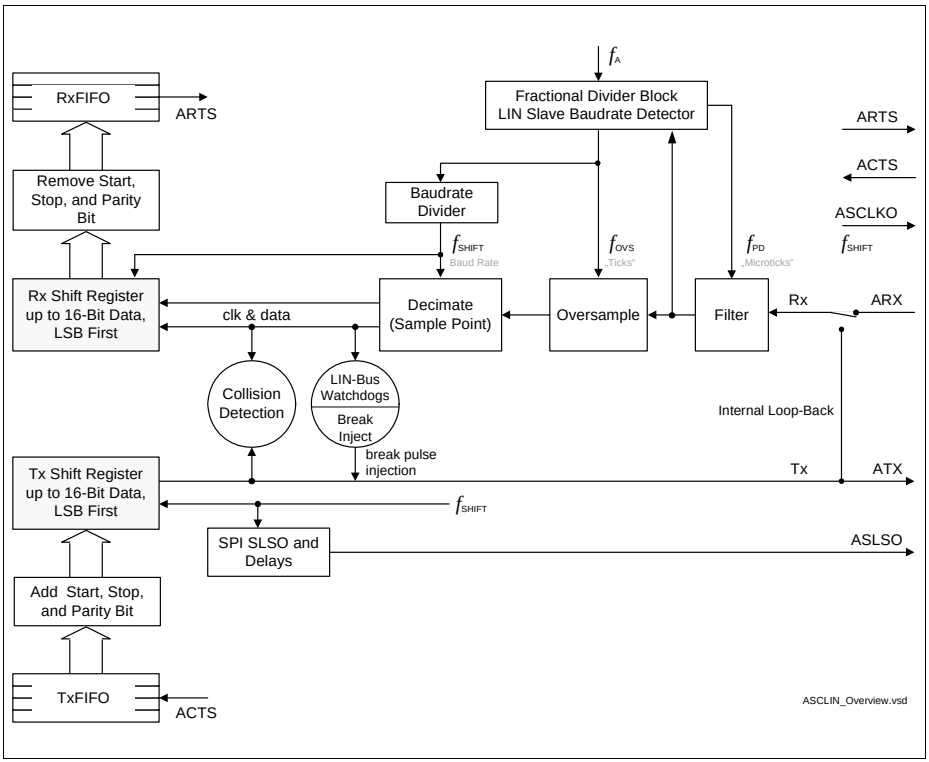


Figure 19-2 Architecture Overview

Note: Collision detection is mandatory only in the LIN specification V2.1.

Note: In LIN mode, both ARX and ATX signal must be connected to the LIN bus in order ASCLIN module to work properly.

19.3 External Signals

The ASCLIN module provides the following external signals:

- Serial clock output ASCLK
- Receive data input ARX (Master Receive MR input in SPI mode)
- Transmit data output ATX (Master Transmit MT output in SPI mode)
- Slave select signal output ASLSO
- Request to send handshake output ARTS
- Clear to send handshake input ACTS

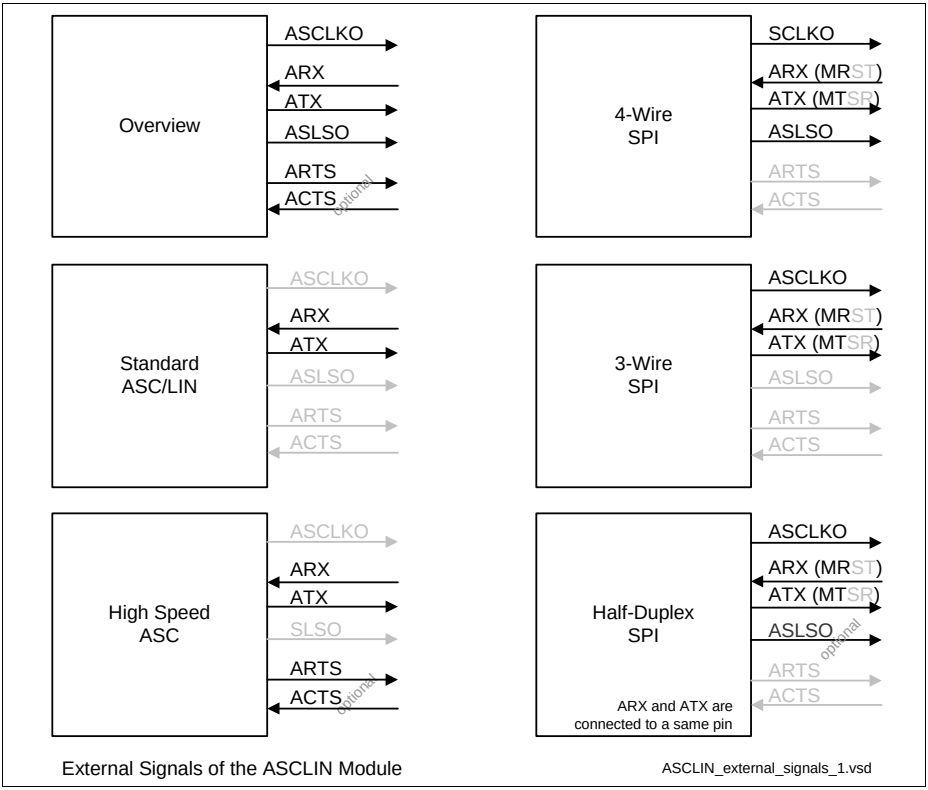


Figure 19-3 External Signals of the ASCLIN Module in Different Modes

19.4 User Interface

The user interface contains a Tx FIFO and an Rx FIFO.

They provide the following services: takes data packets with width optimal for the FPI[1)] bus and packs them to serial frames, buffers the FPI bus data, manages handshaking. These features are designed to optimize the use of the module in LIN, high-speed ASC and SPI mode. They also optimize the use of the module in standard ASC mode.

19.4.1 TxFIFO Overview

The Tx FIFO has the ability to pack 16-bit writes from the FPI bus to two up-to-8-bit frames or one up-to-16-bit frame. It also has the ability to pack 32-bit writes from the FPI bus to four up-to-8-bit or two up-to-16-bit frames.

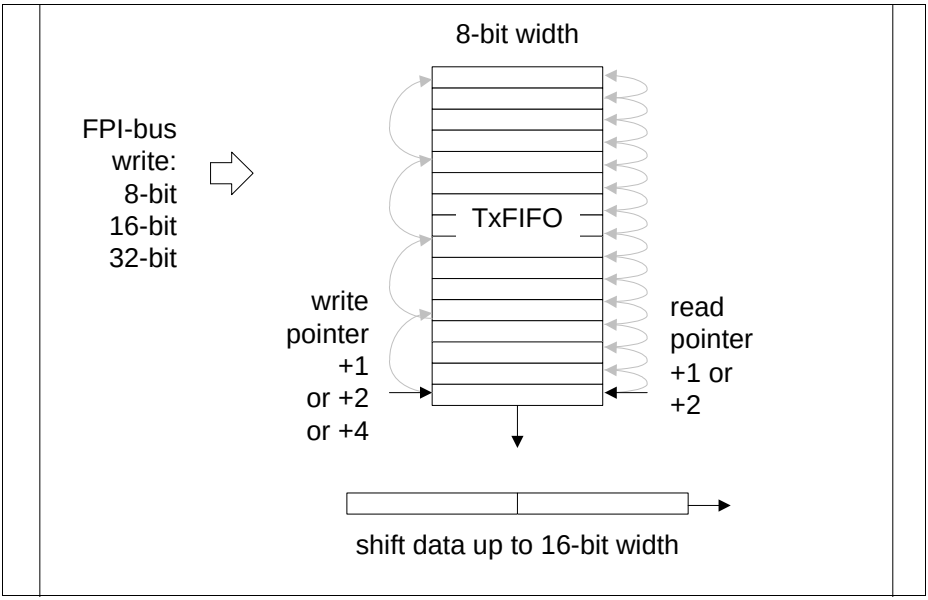


Figure 19-4 TXFIFO Overview

The number of bytes written to the TxFIFO is always defined with the **TXFIFOCON**.INW, does not depend on the SPB write width, and should be equal to it.

If the SPB write width is shorter than INW, then the missing part is padded with zeros.

If the SPB write width is longer then INW, then the excessive part is lost.

1) FPI is the protocol used on the System Peripheral Bus (SPB).

On the shift register side, the TxFIFO is always emptied by a number of bytes as needed for the data width field **DATCON**.DATLEN.

The **Table 19-1** describes all the possible write (fill) combinations. The bytes which will be filled into the TXFIFO are labeled as A, B, C, D and 0.

Table 19-1 Inlet Width versus SPB Write Width

TxFIFO Contents		SPB Bus Write Width		
		8-Bit A	**16-Bit BA**	**32-Bit DCBA**
Inlet Width INW	**8-Bit**	A	A	A
	16-Bit	0A	BA	BA
	32-Bit	000A	00BA	DCBA

19.4.2 Using the TxFIFO

The Tx FIFO has the ability to pack 16-bit writes from the FPI bus to two up-to-8-bit frames or one up-to-16-bit frame.

Note: Some (but not all) use-cases are described in the sections below. The software can always access the TXFIFO 8, 16 or 32 bit wide, according to its requirements.

19.4.2.1 Standard ASC Mode

The most standard usage of the ASC module is to transmit 7 or 8-bit values, and to fill the FIFO with 8-bit wide FPI-bus accesses. The FPI bus access is 8-bit wide, and there is only BS0 (Byte Select 0) signal active.

The transmit data is written to the address TXDATA. The Tx_Inlet_Width is 8-bit, and the Tx_Outlet_Width is 8-bit.

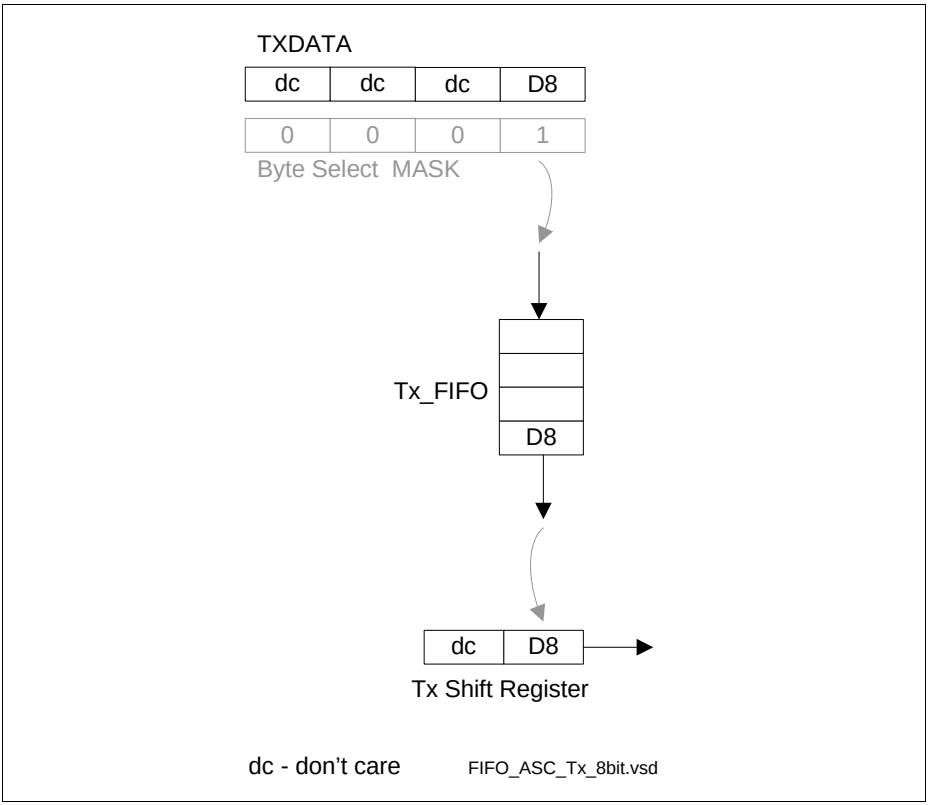


Figure 19-5 FIFO Operation in 7- or 8-Bit ASC Mode, with or without Parity

A less common mode of operation of the ASC module is to transmit 9-bit values, containing either 9-bit data or 8-bit data. The FPI bus access should be 16-bit wide, and there are BS0 and BS1 (Byte Select 0 and 1) signals active. The shift register is filled with two 8-bit values at the location read pointer and read pointer + 1.

The transmit data is written to the address TXDATA. The Tx_Inlet_Width is 16-bit, and the Tx_Outlet_Width is 16-bit.

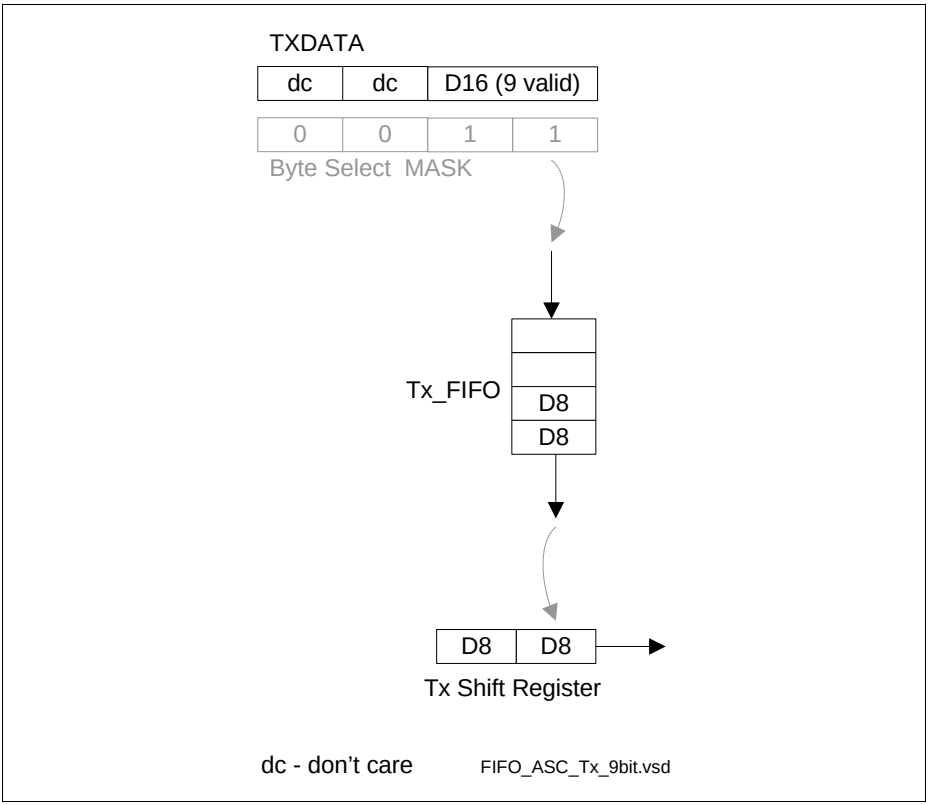


Figure 19-6 FIFO Operation in 9-Bit ASC Mode

19.4.2.2 High Speed ASC Mode

In order to reduce the FPI bus load by not using a 32-bit move for 7 or 8-bit values, one option is to fill the FIFO with 32-bit wide accesses containing four 8-bit wide values. All Byte Select signals BS[3:0] are active, and all four bytes are written in the TX FIFO in one cycle. At the same time, the write pointer of the TXFIFO jumps by the amount of four.

The transmit data is written to the address TXDATA. The Tx_Inlet_Width is 32-bit, and the Tx_Outlet_Width is 8-bit.

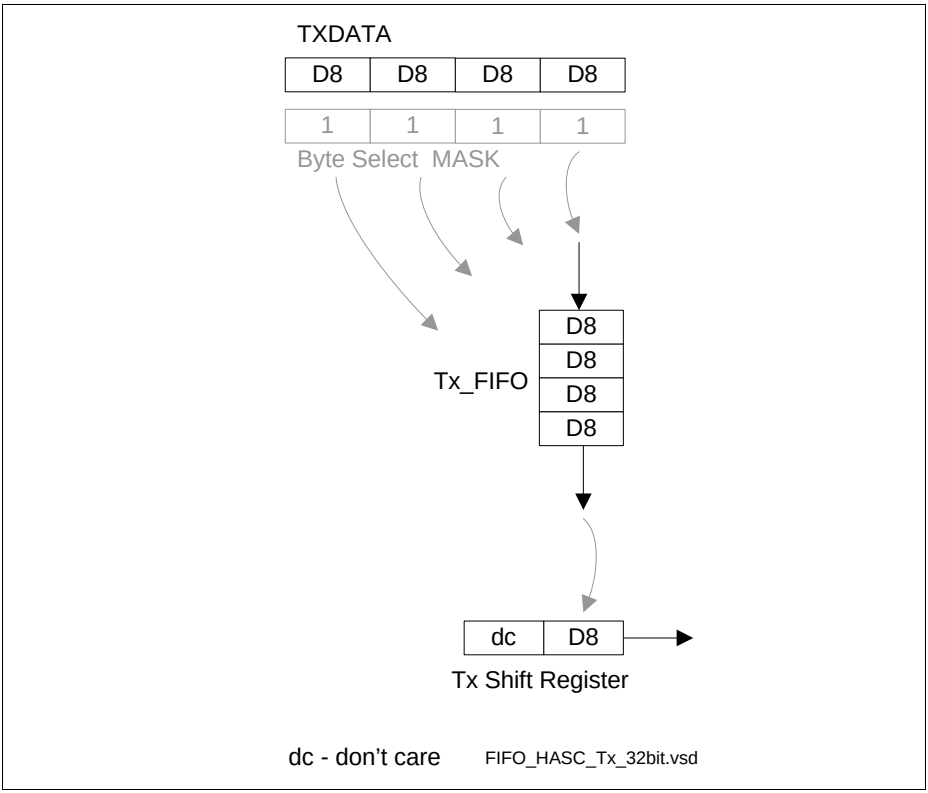


Figure 19-7 High-Speed Communication Scenario

19.4.2.3 LIN Mode

In LIN mode, several 8-bit frames are preceded by a low break pulse.

The generation of the break pulse is programmable with an 8 bit timer in units of bits, where the wide range of break pulses can be generated, for example in a range between 13 and 26 bits and beyond (up to 256 bits, but these lengths would violate the maximum LIN header length). The detection of the break pulse is programmable with an 8 bit timer in units of bits. Among others , the standard thresholds of 10 and 11 bit times can be set. .

The pulse is generated by an module internal timer. The transmit sequence consists of filling the stopped TXFIFO with the appropriate bytes, and then starting the break pulse, which activates the TxFIFO.

The transmit data is written to the address TXDDATA. The Tx_Inlet_Width can be 8-bit (or 16-bit, or 32-bit), and the Tx_Outlet_Width is 8-bit.

19.4.2.4 SPI Mode

SPI mode is used most often to send or receive data of 8 or 16-bits length, or some length in between. Therefore the reading out of the Tx FIFO must be 8 or 16-bit wide, and write could be 16-bit wide, or even 32-bit wide if two frames are packed in one FPI bus access.

The transmit data is written to the address TXDATA. The Tx_Inlet_Width is up to 32-bit, and the Tx_Outlet_Width is up to 16-bit.

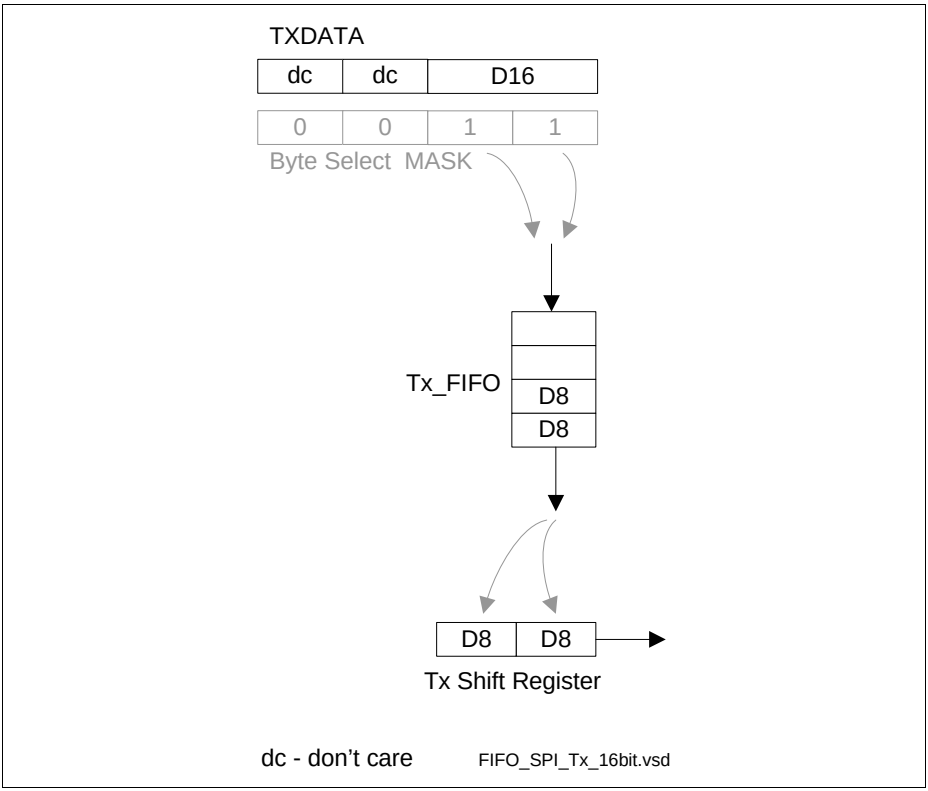


Figure 19-8 TXFIFO Operation in SPI Mode

19.4.3 RxFIFO Overview

The Rx FIFO has the ability to pack two up-to-8-bit frames or one up-to-16-bit frame to one 16-bit write to the FPI bus. It also has the ability to pack four up-to-8-bit 2-up-to-16-bit frames to one 32-bit write to the FPI bus.

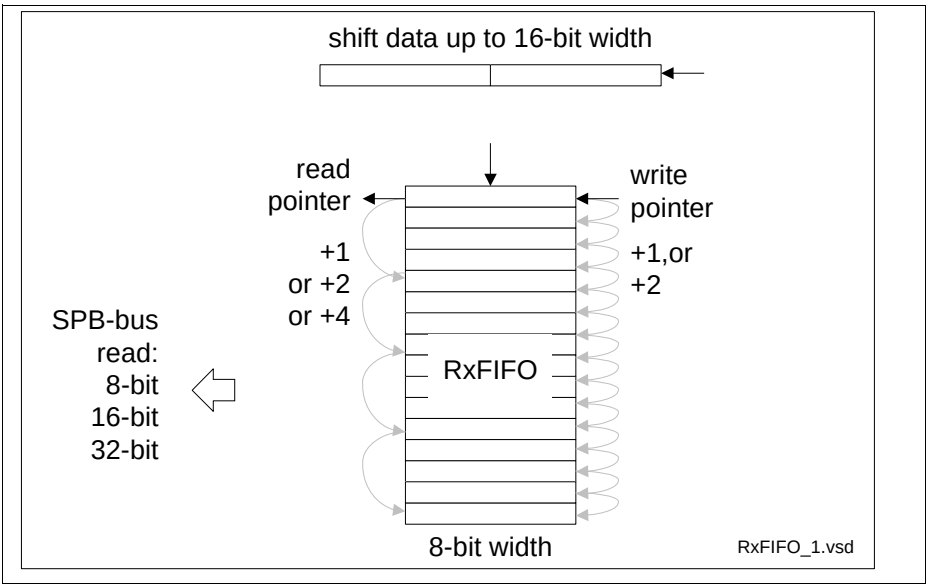


Figure 19-9 RXFIFO Overview

The number of bytes taken out of the RxFIFO is always **RXFIFOCON**.OUTW, does not depend on the SPB bus read width, which should be normally equal to OUTW.

If the SPB read width is shorter than the OUTW, then the missing part is filled with zeros.

If the SPB read width is longer then OUTW width, then the excessive part is lost.

On the shift register side, the RxFIFO is always filled with a number of bytes as needed for the data width field **DATCON**.DATLEN.

The **Table 19-2** describes all the possible combinations. The bytes in the RxFIFO are named A, B, C and D, A being the first one to take out. The padding byte is 0.

*Note: In case of an underflow, i.e. when a read access is performed to RXDATA and the number of bytes to be taken out of the FIFO as given by the setting of **RXFIFOCON**.OUTW exceeds the number of bytes that are actually stored in the FIFO, as indicated by **RXFIFOCON**.FILL, then the RXFIFO delivers the available data padded with zeros up to the read access width. However, the data remain in the FIFO and the filling level **RXFIFOCON**.FILL is not changed. To remove the data, the user software must flush the RXFIFO. The bit **FLAGS**.RFU is set.*

Table 19-2 Outlet Width versus SPB Read Width

SPB-Read Data, taken from the RxFIFO		SPB Bus Read Width		
		8-Bit	**16-Bit**	**32-Bit**
Outlet Width OUTW	**8-Bit**	A	A	A
	16-Bit	0A	BA	BA
	32-Bit	000A	00BA	DCBA

19.4.4 Using the RxFIFO

The Rx FIFO has the ability to pack two up-to-8-bit frames or one up-to-16-bit frame to one 16-bit write to the FPI bus.

19.4.4.1 Standard ASC Mode

The received data is read from the address RXDATA.

In case of data width of 7 or 8 bits, one read on this address delivers one byte and empties the FIFO for one element, if the Rx_Outlet_Width is 8-bit. The Rx_Inlet_Width is 8-bit.

In case of data width of 9 bits, one read on this address delivers two byte and empties the FIFO for two elements, if the Rx_Outlet_Width is 16-bit. The Rx_Inlet_Width is 16-bit.

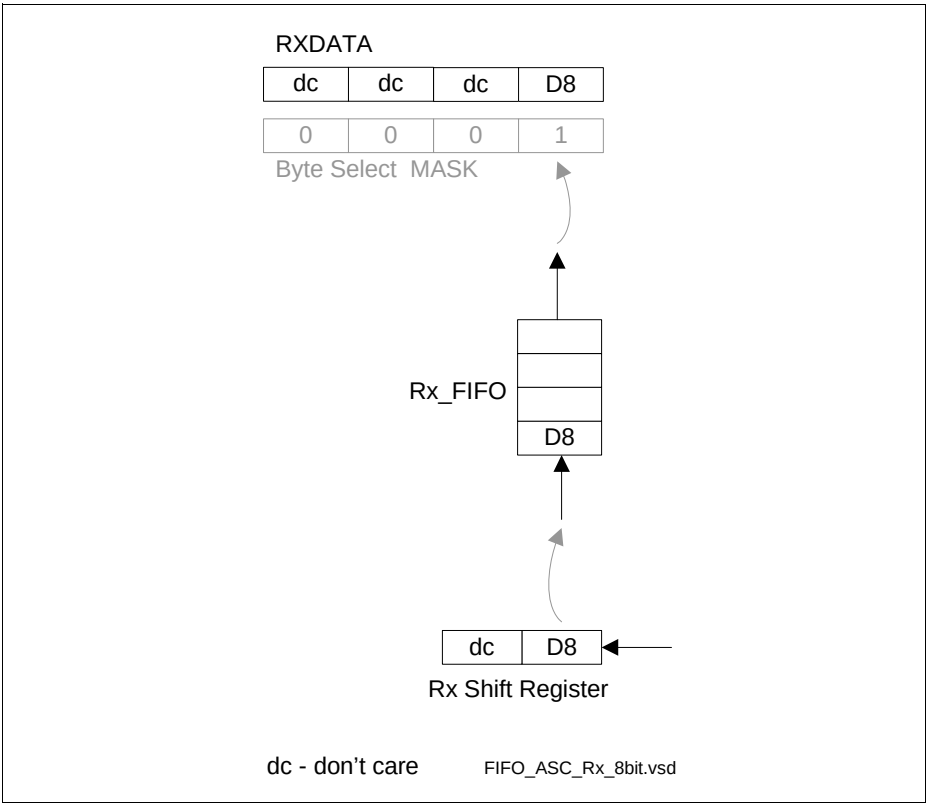


Figure 19-10 8-Bit ASC Reception

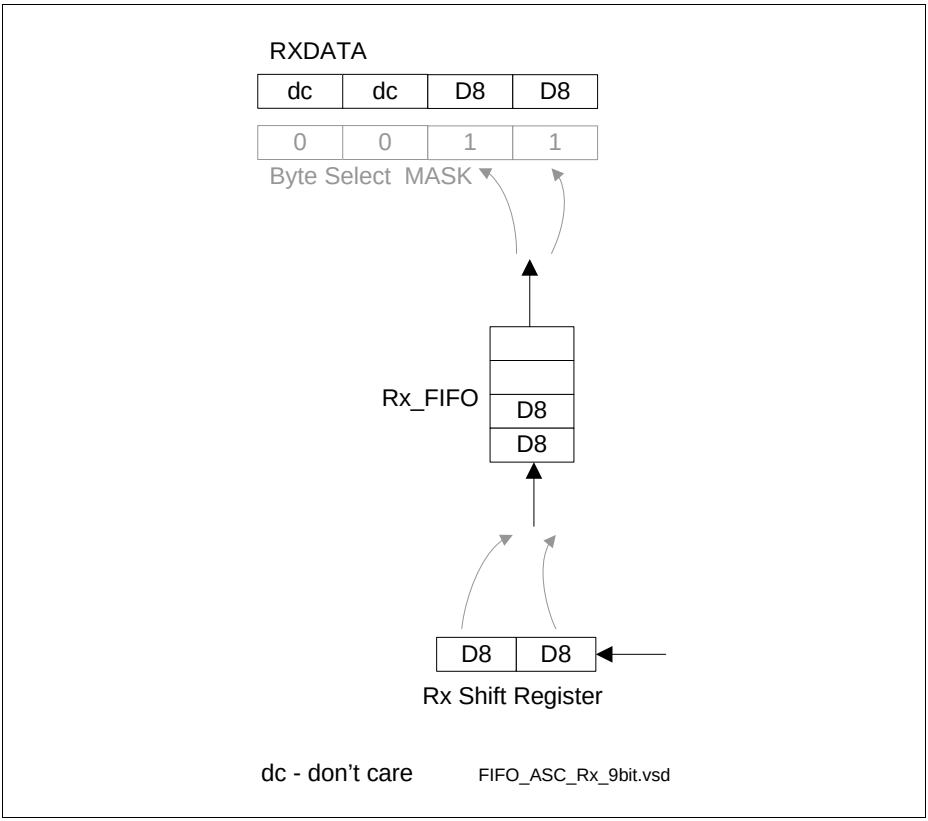


Figure 19-11 9-Bit ASC Reception

19.4.4.2 High Speed ASC Mode

The received data is read from the address RXDATA.

In case of data width of 7 or 8 bits, one read on this address can deliver (one or two or) four bytes and empties the FIFO by four elements, if the Rx_Outlet_Width is 32-bit. The Rx_Inlet_Width is 8-bit.

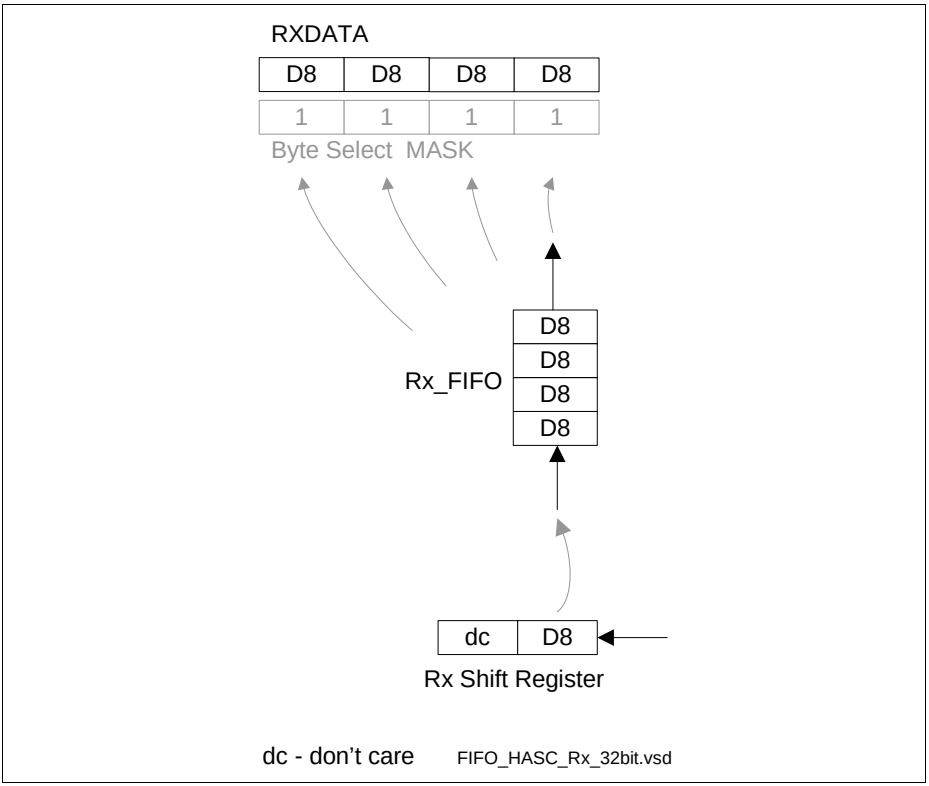


Figure 19-12 High Speed ASC Communication

19.4.4.3 LIN Mode

The received data is read from the address RXDATA.

The read data width is 8 bits (or 16 bit or 32 bit), one read on this address delivers one byte and empties the FIFO by one element, if the Rx_Outlet_Width is 8-bit. The Rx_Inlet_Width is 8-bit

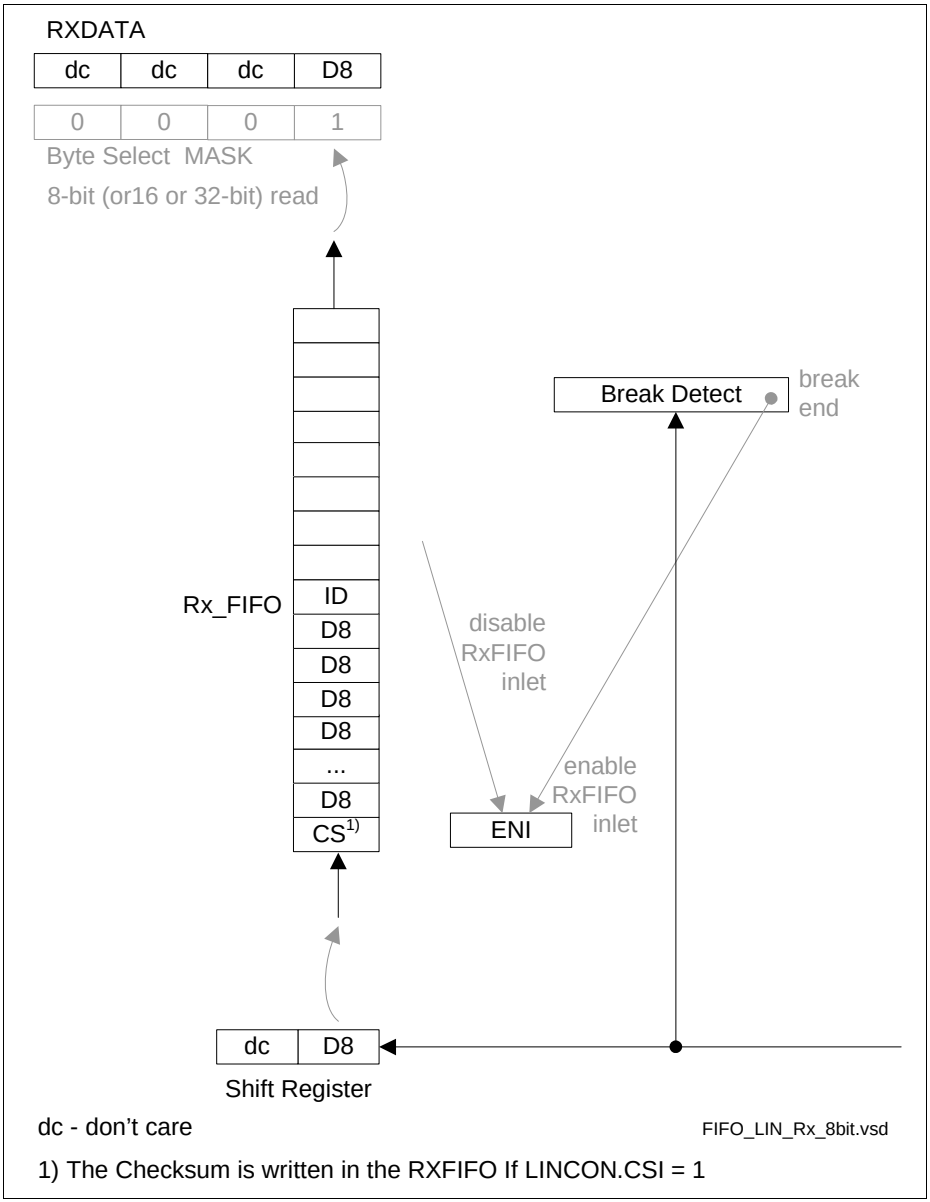


Figure 19-13 RXFIFO Operation in LIN Mode

19.4.4.4 SPI Mode

The received data is read from the address RXDATA.

In case of data width of 16 bits, one read on this address delivers two bytes and empties the FIFO by two elements, if the Rx_Outlet_Width is 16-bit. The Rx_Inlet_Width is 16-bit

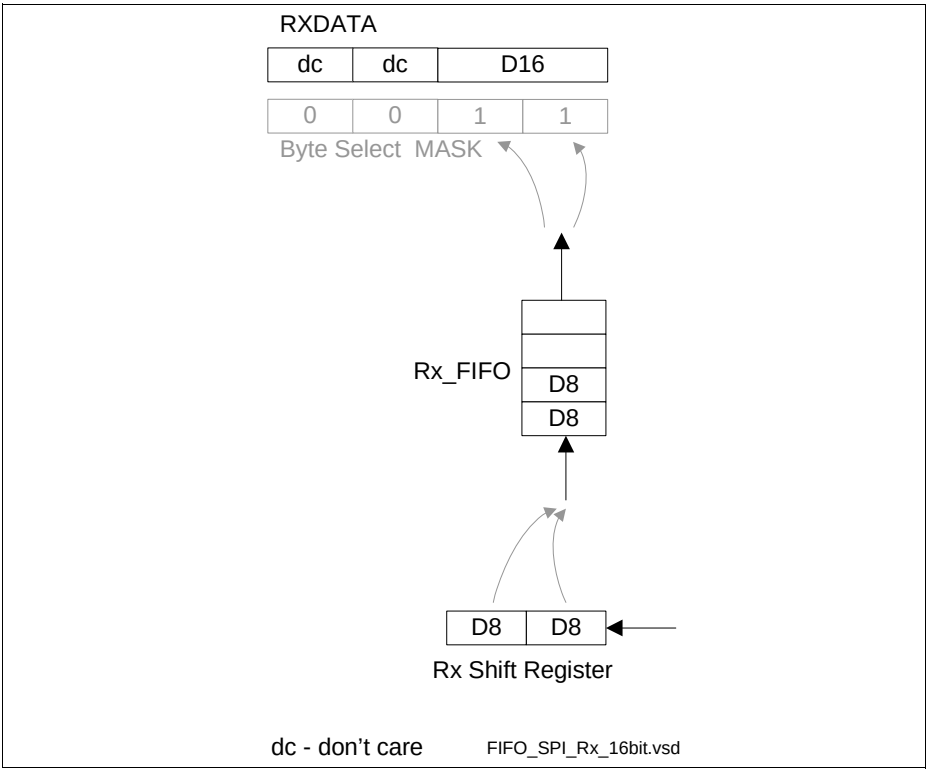


Figure 19-14 RXFIFO Operation in SPI Mode

19.4.5 RTS / CTS Handshaking

A receiver deactivates the RTS (Request to Send) output signal when the RXFIFO is almost full, in order to avoid its overload. When the the emptying of the RXFIFO starts, for example by a DMA, and the filling level falls below the threshold level, the RTS output is activated again. The threshold level is fixed to RXFIFO size minus four, in order to support byte, two bytes and four bytes DMA transfers.

The transmitter receives the RTS output of the receiver on its CTS input and accordingly pauses and resumes the transmission.

19.5 Clock System

The clock system generates all the clocks needed for the proper operation of the ASCLIN module: digital filter clock, oversampling clock, bit time and serial SPI clock.

The clock used for the clock system f_A is independent from the SPB bus clock and remains constant if the SPB bus clock changes, for example in power saving scenarios. The bit field **CSR**.CLKSEL selects the clock source for the f_A frequency, which can be synchronous or asynchronous, higher or lower than the SPB bus frequency.

19.5.1 Baud Rate Generation

Fractional Divider, n-divider and oversampling divider with configurable sample point.

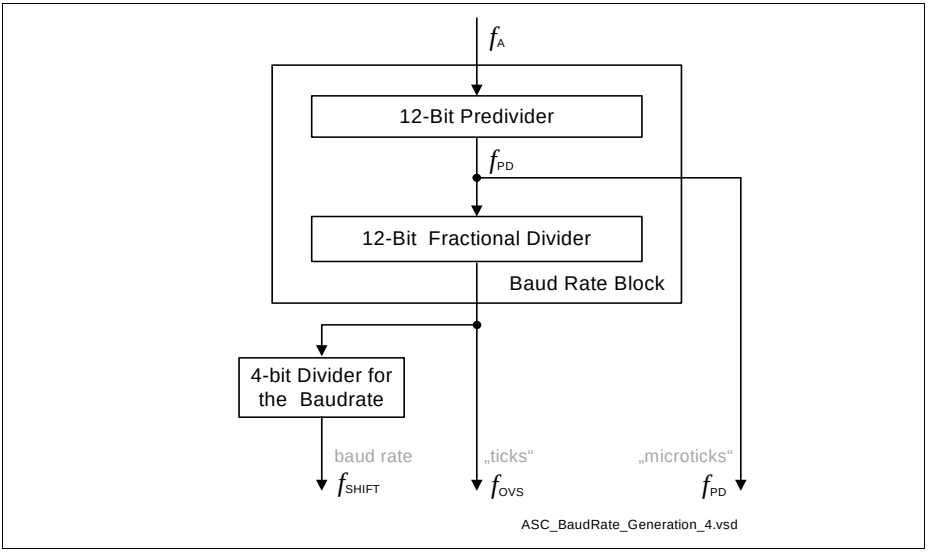


Figure 19-15 Baud Rate Generation

The following bit fields are available for configuring of the 28-bit baud rate divider chain:

- **BITCON**.PRESCALER - the division ratio of the predivider
- **BRG**.NUMERATOR - the nominator of the fractional divider
- **BRG**.DENOMINATOR - the denominator of the fractional divider
- **BITCON**.OVERSAMPLING - the division ratio of the baudrate post divider

The chain of the generated frequencies from f_A down to the f_{SHIFT} (the baudrate) is calculated as follows:

$f_{PD} = f_A$ / (**BITCON**.PRESCALER + 1)

$f_{OVS} = f_{PD}$ * **BRG**.NUMERATOR / **BRG**.DENOMINATOR,

$f_{SHIFT} = f_{OVS}$ / (**BITCON**.OVERSAMPLING + 1)

*Note: Fractional division requires that **BRG**.NUMERATOR is less or equal than the **BRG**.DENOMINATOR.*

The overall formula is given as follows:

(19.1)

$$BAUDRATE = \frac{f_A \times NUMERATOR}{(PRESCALER + 1) \times DENOMINATOR \times (OVERSAMPLING + 1)}$$

19.5.2 Bit Timing Properties

The ASCLIN module provides flexible programming of the bit oversampling, sampling point and input signal filtering properties.

The oversampling factor for the incoming bit-stream is configurable from 4 to 16 ticks (or time quanta) per bit.

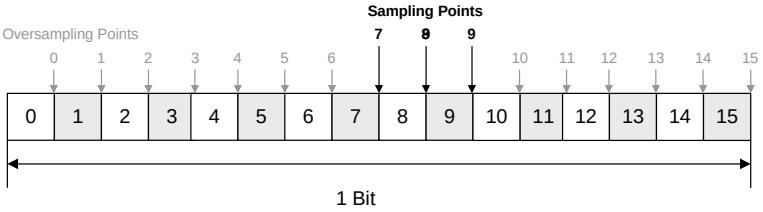
At the same time, using the oversampling frequency, a digital median filter can be enabled to filter the incoming bit stream. The filtering uses the standard majority out of three procedure. If the filter is disabled, then each bit is sampled only once.

The sampling point is also configurable and should be used in conjunction with the oversampling factor. One standard setting is 16x oversampling and using the samples 7, 8 and 9 for the data. Another possible setting could be 8x oversampling and using the samples 3, 4, and 5 for data.

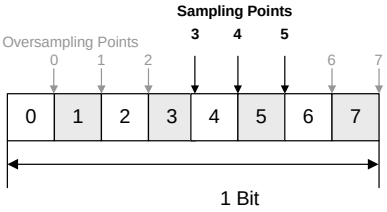
The following bitfields are available for configuring the bit properties:

- **BITCON**.PRESCALER - the twelve bit integer divider defining the microtick used by the fractional divider to generate the baud rate, and by the digital filter for the deglitching of the RX input signal.
- **BITCON**.SAMPLEPOINT - the bit field defining the sampling point position, and the duty cycle in the SPI mode.
- **BITCON**.SM - the bit enables the digital median filter (majority out of three): 1 or 3 samples per bit.
- **IOCR**.DEPTH - the bitfield defining the floating average filter depth: off or 1 to 63 microticks
- **BITCON**.OVERSAMPLING - the bitfield defining the number of ticks per bit, in the range of 4 to 16. This is a post-divider located after the fractional divider.

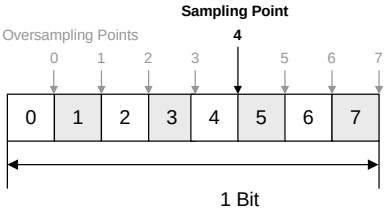
16x Oversampling, 3 sample points, relevant sample position 8



8x Oversampling, 3 sample points, relevant sample position 4



8x Oversampling, 1 sample point, relevant sample position 4



16x Oversampling, 1 sample point, relevant sample position 12

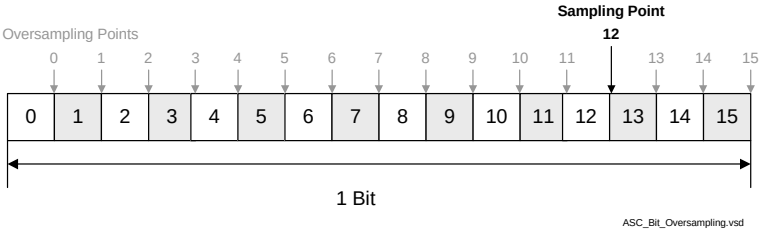


Figure 19-16 ASCLIN Bit Structure

Generally, the sampling point should be placed in the middle of the bit. This position is optimal in case the baud rate (the oscillator frequency) is not very precise and stable.

If the oscillator precision is very high, which is usually the case when two microcontrollers driven by quartz oscillators communicate, but the signal edges are very unsymmetrical, which is the case if open drain half-duplex connection is used, it can be of advantage to move the sample point somewhere in the second half of the bit. Open drain connection usually causes the “0” bits to be longer than the “1” bits. In such a case, optimizing the sampling point would mean placing it in the middle of the shorter “1” bit.

At the end, different combinations of oscillator precision, asymmetry of the edges, and loop-delays for collision detection result in different optimal positions of the sampling point.

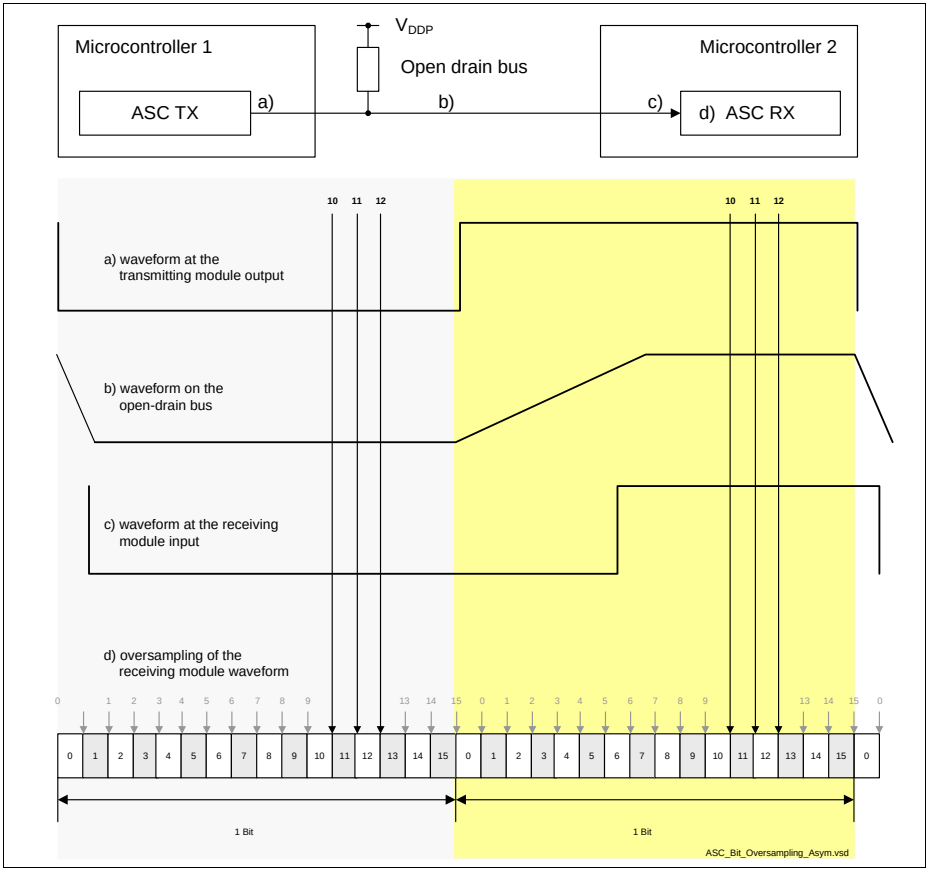


Figure 19-17 Bit Length Asymmetry and the Sampling Point

19.6 Data Frame Configuration

Applicable as transmitter and receiver, the parity scheme is configured in the bit fields **FRAMECON**.ODD, the data length in the bit fields **DATCON**.DATLEN and the stop bits in the bit fields **FRAMECON**.STOP.

19.7 Miscellaneous Configuration

Loop - back

19.8 Synchronous Mode

In synchronous mode the module supports the SPI setting of shift edge first, than the latch edge, see **Figure 19-18**. The module is set in synchronous mode by using the bit field **FRAMECON**.MODE.

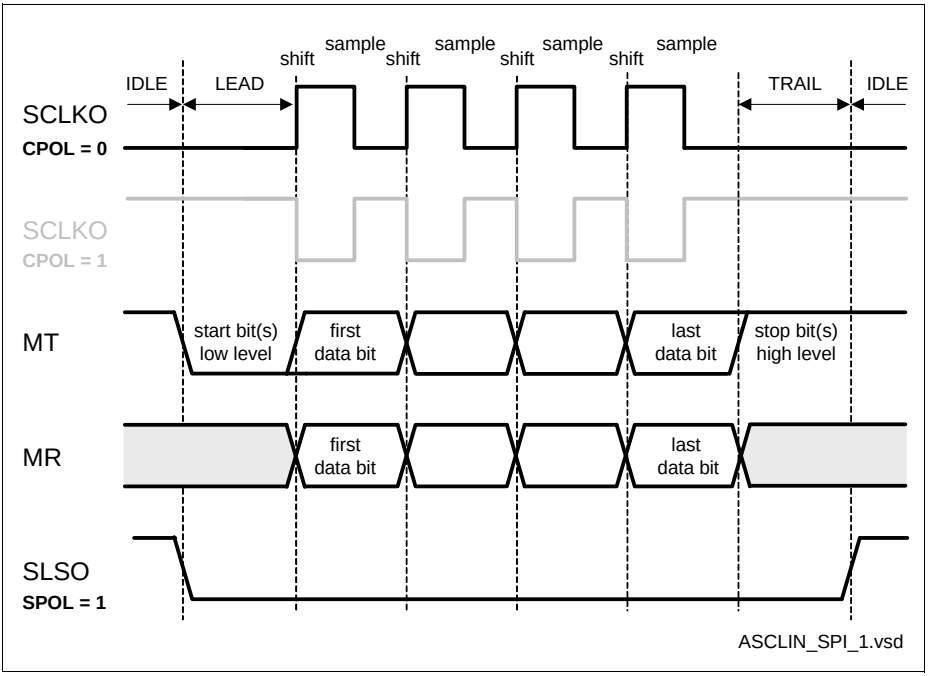


Figure 19-18 SPI Timing

19.8.1 Baud Rate and Clock Generation

The baud rate and clock generation in the synchronous mode uses generally the same counters and principles as in the asynchronous mode. It uses the same prescaler, fractional divider, and oversampling divider with configurable sample point. The only difference is that the shift clock is driven as output signal at the pin SCLKO. If a symmetrical shift clock is required, the oversampling ratio should be an even number, and the sampling point should be set to one half of the oversampling ratio (see the **BITCON** register).

The clock polarity is configured in the bit **IOCR**.CPOL.

19.8.2 Data Frame Configuration

The leading and trailing delays are configured in the bit fields **FRAMECON**.LEAD and **FRAMECON**.STOP. The IDLE phase duration is configured using the bit field **FRAMECON**.IDLE. The data length of 2 to 16 bit is defined in the bit field **DATCON**.DATLEN.

19.8.3 Slave Selects Configuration

The SPI master activates automatically the slave select output signal for each data word. The polarity of the slave select can be configured by using **IOCR**.SPOL

19.8.4 Miscellaneous Configuration

Loop - back

19.9 LIN Support

The ASCLIN module provides hardware support for the LIN protocol.

It supports all four elementary LIN transactions:

- TxH - Transmission of Header
- TxR - Transmission of Response
- RxH - Reception of Header
- RxR - Reception of Response

By supporting additionally the combinations of these elementary transactions, the module actually supports all LIN use cases: sending and receiving headers and responses as

- LIN master or
- LIN slave

A LIN master is engaged in three elementary transactions: TxH, TxR, RxR. It never engages in RxH, because a master never receives a header, it only transmits a header.

A LIN slave engages also in three elementary transactions: RxH, TxR, RxR. It never engages in TxH, because a slave never transmits a header, it only receives a header.

Each elementary transaction needs some hardware resources in order to be completed with minimum CPU intervention. Here is a list of tasks per transaction, supported by hardware, and the required hardware resources:

- TxH - Transmission of Header - master mode only
  - break generation: 8-bit bit-field defining the break length in units of bits
  - sync-field generation: hard coded 55H byte
  - ID transmission with interrupt generation
- TxR - Transmission of Response - master and slave mode
  - number of bytes parameter: bit-field of length 4
  - checksum generation: hardware engine, supporting classic and enhanced checksum, which can be enabled or disabled
- RxH - Reception of Header - slave mode only
  - optional auto-baud detection: fractional divider with programmable nominator and denominator
  - number of bytes parameter: bit-field of length 4
  - checksum detection: hardware that supports classic and enhanced checksum, which can be enabled or disabled; a checksum error is flagged, and an error interrupt can be triggered, if enabled
  - interrupt at end of header (necessary to set the number of bytes for the RxR phase)
  - timeout on overflow: 8-bit timer
  - break detection:8 bit timer with programmable threshold in units of bits
- RxR - Reception of Response - master and slave mode
  - number of bytes parameter: bit-field of length 4

– checksum: the received checksum is optionally delivered in the RXFIFO
– timeout on overflow: 8-bit timer

The break detection feature is always active.

Wake-up signal generation in both slave and master mode

For many LIN configuration parameters, the hardware of the ASCLIN module provides wider ranges than the LIN standard parameters. Therefore the application software shall take care that appropriate LIN standard values are used to configure the module. Such configuration parameters are:

- break length
- data width
- break threshold
- wake-up length
- header, frame and response timeout
- idle time

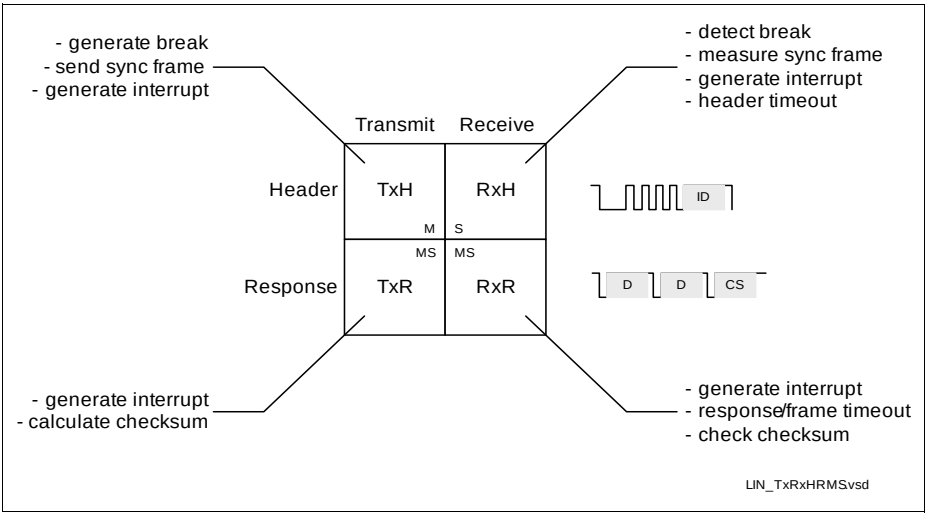


Figure 19-19 Overview of the elementary LIN transactions

19.9.1 LIN Watchdog

The LIN watchdog monitors the duration of the header, the frame or the response, and appearance of break pulses. It checks against the pre-defined time limits.
If the limits are violated, timeout interrupts are generated.

The LIN watchdog is necessary in slave mode.

The wake pulse generation is performed using the shift register and an appropriate data byte containing several consecutive zero bits. The wake pulse detection is done using falling edge detection.

For monitoring the bus for long idle or zero states flags for rising and falling edge are available. These flags can be polled with some appropriate time raster (in microseconds or miliseconds range).

The header timeout value is known at module initialization time and remains constant for all frames.

The frame or response timeout value depends on the length of the response (1 to 8 bytes) and must be set by software depending on the received ID, after the header has been received. The initial value of this timeout should be set by software to the maximum allowed by the timer, that is 256.

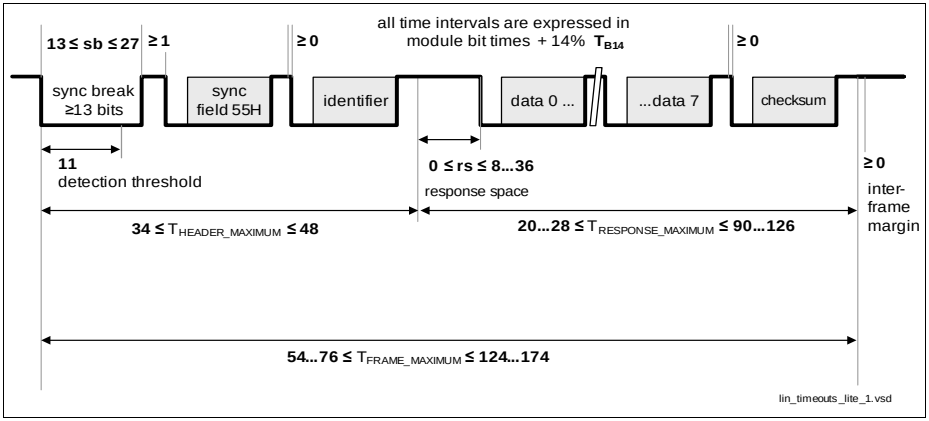


Figure 19-20 Duration of the Elements of a LIN Frame

The module provides timer blocks running in parallel, performing watchdog functions on specific timing requirements of the LIN protocol.

19.9.1.1 LIN Break, Wake, Stuck Handling

This subsection describes:

- the monitoring of the LIN Bus
- Break Pulse detection and generation and
- Wake Pulse detection and generation.

Monitoring the Bus

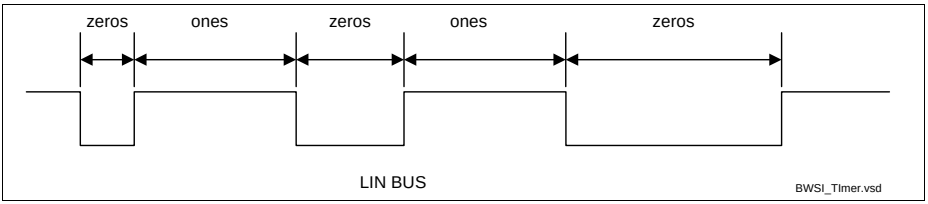


Figure 19-21 LIN Bus View as Sequence of Sequences of Zeros and Ones

Monitoring the bus for long idle periods (or stuck periods) is supported by providing the bit fields **FLAGS**.RED and **FLAGS**.FED. Polling these fields periodically within some time raster defined by the operating system or timer for example each 1ms or each 10 ms can be used for detecting very long inactive periods in the range, for example, of 150 ms to 10s. Additionally, an interrupt on edge can be enabled by using the corresponding enable bits in the **FLAGSENABLE** registers. Setting and clearing the flags can be done by software by using the **FLAGSSET** and **FLAGSCLEAR** register.

Note: The low break or wake up pulses do not cause a dummy byte to be transferred to the receive FIFO and do not cause a frame violation error interrupt. After the pulse the shift register goes into initial state and waits for the falling edge of the start bit of the next frame.

Break and Wake Pulse Detection

The detection of the break pulse, which is defined as low pulse with a minimum duration of 13 bit times, is performed using a threshold of either 10 or 11 bit times. The break detection threshold is set using the bitfield **LINBTIMER**.BREAK.

Detecting a break pulse anywhere in the frame resets the LIN state machine to the “Break Delimiter Detected” state BDD and the whole sequence inclusive the watchdog timeout counting starts from the beginning .

Note: Detecting a break pulse anywhere in the frame could result in setting the Frame Error Flag (FE)

Wake low-pulse detection is done by monitoring the bus for a falling edge in sleep mode. If an early wake-up shall be suppressed, the parameter **IOCR**.DEPTH (glich filter) may be used.

Break and Wake Pulse Generation

The generated break low-pulse duration is defined by the bit field **LINBTIMER**.BREAK in the range of 1 to 64 bits.

Wake low-pulse is nominal 5 bit times long, but can be set to any value between 1 and 9, by writing the appropriate character in the TXFIFO and requesting its transmission as the wake pulse by setting **FLAGS**.TWRQ.

Note: Injecting the low pulse disables the reception with the shift register, so that this low pulse is not detected or treated as a normal ASC frame.

19.9.1.2 LIN Header and Response Timers

The LIN header and response times can be monitored separately.

(>>**LINHTIMER**, **DATCON** description)

- Header duration measurement
  - In master mode starting point of the time measurement is falling edge of the break pulse.
  - In slave mode starting point of the time measurement is the moment of detecting that there is a break pulse going on, that is the moment of detection of a low pulse longer than either 10 or 11 bit times, as configured in the **LINBTIMER** register.
- Response duration measurement
  - In both master and slave modes the response duration measurement starts with the end of the header and ends with the end of the response. “End” means end of the last bit of the last byte (of the checksum byte).

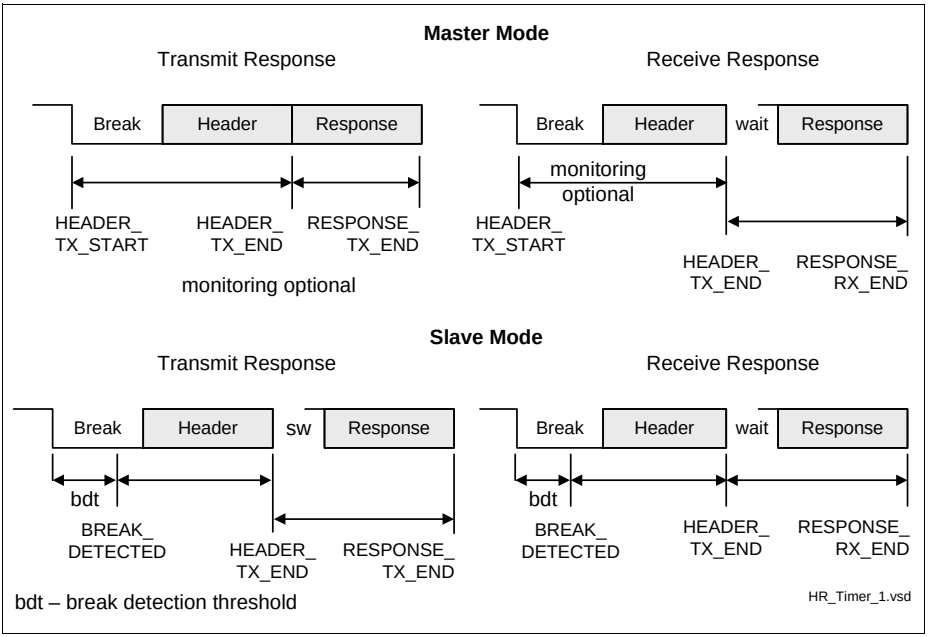


Figure 19-22 Duration Measurement of the Header and Response

Note: Transmitting master node can optionally monitor its own header or response timouts, in order to detect some error conditions, like TXFIFO not containing the ID or data to be transmitted. However, these error conditions can be detected also in other ways.

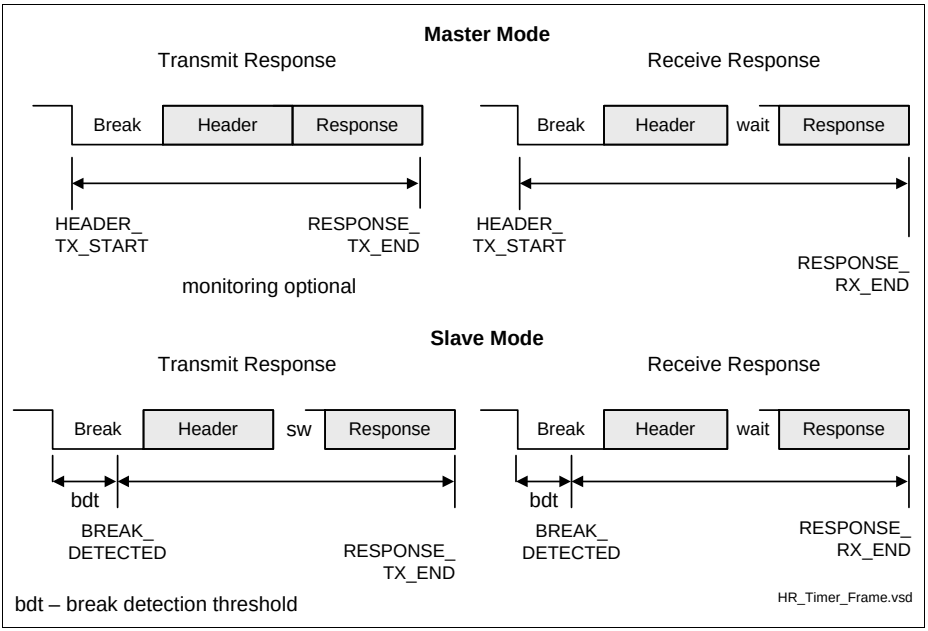


Figure 19-23 Duration Measurement of the Response or Frame

Acording to the setting of the bit **DATCON**.RM, the **DATCON**.RESPONSE bit field defines response or frame duration threshold.

19.9.2 LIN Master Sequences

The sequences described below are examples how the elementary LIN transactions can be executed. There exist other ways to do them, for example by using other events to control the protocol flow: fifo level events instead of end of header / response events. Some alternatives are indicated in the lists below with brackets.

Initialize the module in LIN master mode:

- deactivate the clock source : **CSR**.CLKSEL = 0 (off)
- wait or poll for **CSR**.CON = 0 (see also **Clock Reconfiguration**)
- enter the INIT mode : **FRAMECON**.MODE = INIT
- activate the clock source : set **CSR**.CLKSEL
- wait or poll for **CSR**.CON = 1
- deactivate the clock source : **CSR**.CLKSEL = 0 (off)
- wait or poll for **CSR**.CON = 0
- select the operation mode : **FRAMECON**.MODE = 3 (LIN)
- configure the master mode : **LINCON**.MS = 1
- configure the baud rate : **BRG**
- configure the bit timing : **BITCON**.PRESCALER and OVERSAMPLING
- configure the bit sampling : **IOCR**.DEPTH, **BITCON**.SM, SAMPLEPOINT
- configure the frame parameter : **FRAMECON**.PEN=0 (no parity), STOP = 1
- configure the break length : **LINBTIMER**.BREAK = 13 (typ)
- configure the delay parameter : **FRAMECON**.LEAD, **FRAMECON**.IDLE
- configure the checksum mode : **LINCON**.CSEN, **LINCON**.CSI
- activate the clock source : set **CSR**.CLKSEL
- wait or poll for **CSR**.CON = 1

Send only a header [master task]:

- configure the watchdog timer : **DATCON**.RESPONSE =256 (max)
- configure the RXFIFO : **RXFIFOCON**.ENI = 0 (or 1), OUTW, BUF=0, flush
- configure the TXFIFO : **TXFIFOCON**.ENO = 0, INW, flush
- clear the event flags : **FLAGSCLEAR**.THC, TRC, RRC, CEC, FEC, HTC, RTC, LPC, LC, TFOC/LC, RFOC/UC/LC
- enable the TX interrupt event : **FLAGSENABLE**.THE
- enable the EX interrupt events : **FLAGSENABLE**.HTE, CEE, (FEE, LPE)
- write into the TXFIFO : **TXDATA** = ID byte
- start the header transmission : **FLAGSSET**.THRQS = 1
- configure the TXFIFO : **TXFIFOCON**.ENO = 1

Note: If the software needs the sent ID in the RXFIFO, it should add the setting of ENI=1 to the sequence above - before triggering the header transmission. In such a case

after the header has been transmitted the ID byte will be available in the RXFIFO, but not the sync byte.

React to the interrupt(s) indicating the end of transmission (and reception) of the ID byte:

- check the error flags : **FLAGS**.HT, CE (FE, LP)
- read from the RXFIFO : **RXDATA**=received ID, if no error has been detected

Check the transmitted / received ID to determine with which response sequence: transmit, receive or ignore to continue (same procedure as in the slave mode).

Send a response to the latest header [slave task]:

- configure the response count : **DATCON**.DATLEN
- configure the watchdog timer : **DATCON**.RESPONSE
- configure the checksum mode: **DATCON**.CSM
- configure the RXFIFO : **RXFIFOCON**.ENI =0
- configure the TXFIFO : **TXFIFOCON**.ENO = 1
- clear the event flags : **FLAGSCLEAR**.TRC, TFOC/UC, CEC, RTC, BDC
- enable the TX interrupt event : **FLAGSENABLE**.TRE
- enable the EX interrupt events: **FLAGSENABLE**.TFOE, CEE (RTE), BDE
- write into the TXFIFO : **TXDATA** = data bytes D0, D1, ... (and optionally the checksum byte if the hardware checksum generation is disabled)
- start response transmission : **FLAGSSET**.TRRQS = 1
- the corresponding interrupts signal an error or the end of the response transmission

Receive a response to the latest header [slave task]:

- configure the response count : **DATCON**.DATLEN
- configure the watchdog timer : **DATCON**.RESPONSE
- configure the checksum mode: **DATCON**.CSM
- configure the RXFIFO : **RXFIFOCON**.ENI = 1
- clear the event flags : **FLAGSCLEAR**.RRC, RFOC/UC, FEC, LCC, RTC, FEDC, REDC, BDC
- enable the RX interrupt event : **FLAGSENABLE**.RRE
- enable the EX interrupt events: **FLAGSENABLE**.RFOE/UE, FEE, (LCE), RTE, BDE
- The corresponding interrupts signal that the whole response has been received (= end of the frame) or an error has occured.

Note: If checksum injection has been enabled, the received (custom) checksum is also written into the RXFIFO and should be taken into account.

- Fetch from the RXFIFO : **RXDATA** = received data, if no error occured

*Note: The hardware checksum check can be enabled using the bit **LINCON**.CSEN. The received checksum write to the RXFIFO can be enabled using the **LINCON**.CSI.*

Ignore the latest header [slave task]:

- configure the RXFIFO : **RXFIFOCON**.ENI = 0
- set the header only mode : **DATCON**.HO = 1
- clear the event flags : **FLAGSCLEAR**.FEDC, REDC, BDC

The LIN master always knows if the header is followed by a response transmission (by himself) or a response reception (by a slave) or the response transmission is from slave to slave, except in case of a slave that does not respond always, but driven by internal events.

In case of addresing an always responding slave, it is recommended first to configure the header and response (transmission or reception), and afterwards to start the header transmission and optionally, at the same time, the response transmission.

To start a transaction consisting of sending a header and sending or receiving a response, which makes one whole LIN frame [master node]:

- configure the response count : **DATCON**.DATLEN
- configure the watchdog timer : **DATCON**.RESPONSE
- configure the checksum mode : **DATCON**.CSM
- configure the RXFIFO : **RXFIFOCON**.BUF = 0, flush, ENI = 0
- configure the TXFIFO : **TXFIFOCON** flush, ENO = 0
- clear the interrupt event flags : **FLAGSCLEAR**.THC, TRC, RRC, CEC, FEC, HTC, RTC, LPC, LCC, TFOC/UC/LC, RFOC/UC/LC, FEDC, REDC, BDC
- send case
  - enable the TX interrupt event : **FLAGSENABLE**.TRE, (THE, HTE)
  - enable the EX interrupt events: **FLAGSENABLE**.TFOE, CEE, (FEE, RTE, LPE, BDE)
- receive case
  - enable the RX interrupt event: **FLAGSENABLE**.RRE. (THE, HTE)
  - enable the EX interrupt events: **FLAGSENABLE**.RFOE/UEE, CEE, FEE, RTE, (LCE, BDE)
- write into the TXFIFO : **TXDATA** = ID byte
- send case
  - write to TXFIFO : **TXDATA** = data bytes (and optionally checksum byte if the hardware checksum is disabled)
- start the header transmission : **FLAGSSET**.THRQS = 1
- configure the TXFIFO : **TXFIFOCON**.ENO = 1

Note: If the software needs the sent ID in the RXFIFO, it should add the setting of ENI=1 to the sequence above - before triggering the header transmission. In such a case after the header has been transmitted the ID byte will be available in the RXFIFO, but not the sync byte.

- send case
  - start transmit response : also set **FLAGSSET**.TRRQS = 1

- the corresponding interrupt signals an error or the end of the response transmission
- read from the RXFIFO : if no error detected, **RXDATA** = received ID
- receive case
  - fetch from the RXFIFO : if no error has been detected, **RXDATA** = received data bytes and optionally the checksum byte

19.9.3 LIN Slave Sequences

The sequences described below are examples how the elementary LIN transactions can be executed. There exist other ways to do them, for example by using other events to control the protocol flow: fifo level events instead of end of header / response events. Some alternatives are indicated in the lists below with brackets.

Initialize the module in slave mode:

- deactivate the clock source : **CSR**.CLKSEL = 0 (off)
- wait or poll for **CSR**.CON = 0 (see also **Clock Reconfiguration**)
- enter the INIT mode : **FRAMECON**.MODE = INIT
- activate the clock source : set **CSR**.CLKSEL
- wait or poll for **CSR**.CON = 1
- deactivate the clock source : **CSR**.CLKSEL = 0 (off)
- wait or poll for **CSR**.CON = 0
- select the operation mode : **FRAMECON**.MODE = 3 (LIN)
- configure the slave mode : **LINCON**.MS = 0
- configure the baud rate : **BRG**, autobaud detection enable, **BRD**.UPPER / LOWERLIMIT
- configure the bit timing : **BITCON**.PRESCALER and OVERSAMPLING
- configure the bit sampling : **IOCR**.DEPTH, **BITCON**.SM, SAMPLEPOINT
- configure the frame parameter : **FRAMECON**.PEN = 0 (no parity), STOP= 1
- configure the break length : **LINBTIMER**.BREAK = 11 (typ)
- configure the header timeout : **LINHTIMER**.HEADER
- configure the delay parameter : **FRAMECON**.IDLE
- configure the checksum mode : **LINCON**.CSEN, **LINCON**.CSI
- activate the clock source : set **CSR**.CLKSEL
- wait or poll for **CSR**.CON = 1

In slave mode, the module waits for a header from the master, that is, it waits for a break pulse followed by the sync byte and the ID.

Configure for header reception [slave task]:

- configure the watchdog timer : **DATCON**.RESPONSE =256 (max)
- configure the RXFIFO : **RXFIFOCON** bufmode = 0, flush, ENI = 0
- clear the interrupt flags : **FLAGSCLEAR**.RHC, TRC, RRC, FEC, HTC, RTC, BDC, LPC, LAC, LCC, TFOC/UC/LC, RFOC/UC/LC
- enable the RX interrupt event : **FLAGSENABLE**.RHE
- enable the EX interrupt event : **FLAGSENABLE**.HTE, CEE, FEE, LAE, LPE

React to the interrupt(s) generated after receiving the ID byte:

- check error flags : **FLAGS**.HT, FE, LA, LP
- read from the RXFIFO : if no error detected, **RXDATA** = received ID

Check the received ID to determine which response sequence (send, receive or ignore header) to use next.

The LIN slave does not know in advance if it will respond to the header with a transmission himself or it will receive a response by another slave. It looks up the received ID and decides if this ID is associated with a transmit or receive response or if it is don't care (it is for some other slave). This look-up and the subsequent configuration of the module must be performed by software within the allowed response time.

Send a response to the latest header [slave task]:

(same sequence as in master mode)

Receive a response to the latest header [slave task]:

(same sequence as in master mode)

Ignore the latest header (if the received ID contains an error or is not for this slave) [slave task]:

(same sequence as in master mode)

19.9.4 Using the ENI and HO Bits

Both the ENI (Enable Input) and HO (Header Only) bits affect the reception of the various byte types in the LIN frame.

The HO bit is normally used in cases where the response part of a frame should be ignored; the module waits for the next break signal. The ENI bit can be set by the user software, but is also set by the hardware in slave mode after the sync byte reception, in order to enable the transfer of the ID byte of the header in the RXFIFO.

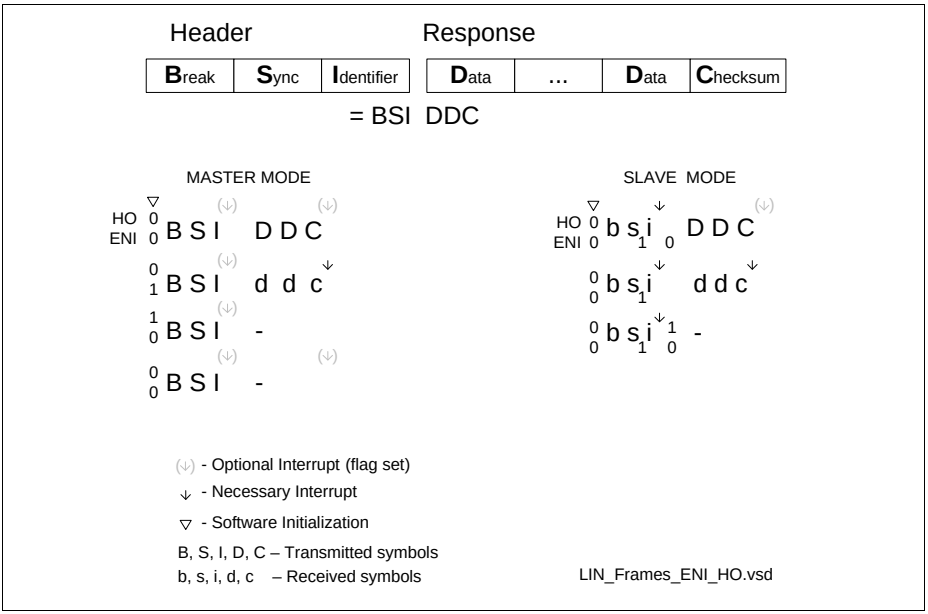


Figure 19-24 ENI and HO bits

19.9.5 LIN Error Recovery

This section describes the behavior of the module in case of errors detected during reception and during transmission of a LIN frame.

Recepton Related Errors

In case of an reception error, the receive state machine goes into the state for waiting for break, and the corresponding interrupt is triggered.

ID Parity error

Checksum error

Timeout error

Framing error

Baud Rate error

Transmission Related Errors

Collision error (LIN2.1 mandatory). If the collision detection mechanism is enabled **FLAGSENABLE**.CEN, the frame will be aborted and the transmiter state machine goes to the idle state.

19.9.6 LIN Sleep and LIN Wake-Up

Wake up low pulse in duration of 250us to 5ms can be generated by the module.

Wake up low pulse in duration longer than 150us wakes up a sleeping module.

A master node which has received a wake up pulse can start polling the slaves, that is it can start sending break pulses and headers, and sending or receiving corresponding responses.

A slave which has been woken up shall be capable of receiving LIN headers after a wake-up time of maximum 100ms. A slave issuing a wake-up pulse expects to receive a header within 150ms to 250ms after the end of the wake-up pulse. If the header does not come, the slave issues a wake up pulse again.

19.10 Auto Baud Rate Detection

Auto baud rate detection is active in slave mode during the reception of the sync field in the LIN header. It measures the longest time interval between two falling edges in the 55H sync field. The measured value is loaded in the denominator of the fractional divider and used afterwards for generating the baudrate for the reminder of this frame if the auto baud rate usage is enabled (**LINCON**.ABD = 1).

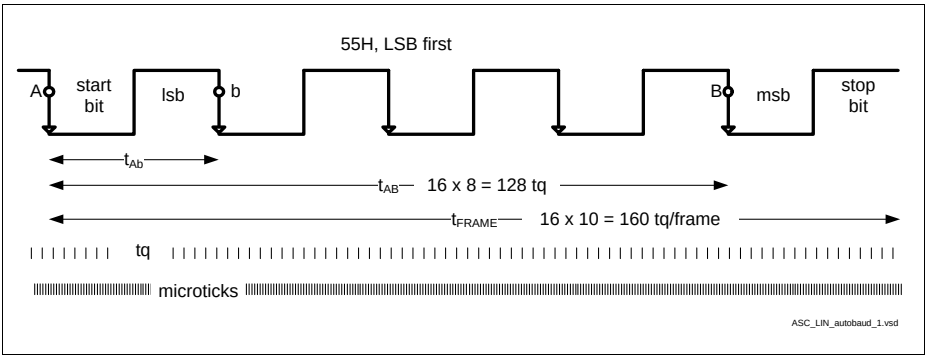


Figure 19-25 Measurement of the Sync field

The following bitfields are available for monitoring the autobaud detection:

- **BRD**.MEASURED - the measured time interval between the first and the fifth falling edge of the sync byte
- **BRD**.UPPERLIMIT - In case the LIN autobaud detection measures a baud rate 14% lower than the nominal one, that is measures a time interval longer than the UPPERLIMIT, a baud rate error event is triggered that, if enabled, generates an interrupt.
- **BRD**.LOWERLIMIT - In case the LIN autobaud detection measures a baud rate 14% higher than the nominal one, that is measures a time interval shorter than the LOWERLIMIT, a baud rate error event is triggered that, if enabled, generates an interrupt.

Auto Baud Rate Operation

The **BRD**.UPPERLIMIT defines the maximum allowed duration for 8 bits in microticks, and therefore the minimum allowed baud rate. In order to define 14% lower baud rate, 8-bit duration 16% longer than the nominal must be entered in the UPPERLIMIT bit field. This is due to the fact that BaudRate = 1 / BitTime, and UPPERLIMIT defines the time.

The **BRD**.LOWERLIMIT defines the minimum allowed duration for 8 bits in microticks, and therefore the maximum allowed baud rate. In order to define 14% higher baud rate, 8-bit duration 12% shorter than the nominal must be entered in the LOWERLIMIT bit

field. This is due to the fact that BaudRate = 1 / BitTime, and LOWERLIMIT defines the time.

If the autobaud is activated, the fractional divider uses a numerator value of 8 x (**BITCON**.OVERSAMPLING + 1) and ignores the bit field **BRG**.NUMERATOR. For the standard LIN protocol oversampling of 16 (**BITCON**.OVERSAMPLING = 15), the numerator value used internally is 128, ignoring the bit field **BRG**.NUMERATOR. Nevertheless, programming 128, or 8 x (OVERSAMPLING + 1) in **BRG**.NUMERATOR may increase the clarity of the software.

Regarding the denominator of the fractionial divider, defined in **BRG**.DENOMINATOR, its initial value is the nominal value and is set by the application software. During the operation of the module, the **BRD**.MEASURED value is automatically loaded into the **BRG**.DENOMINATOR, as long as it is within the limits.

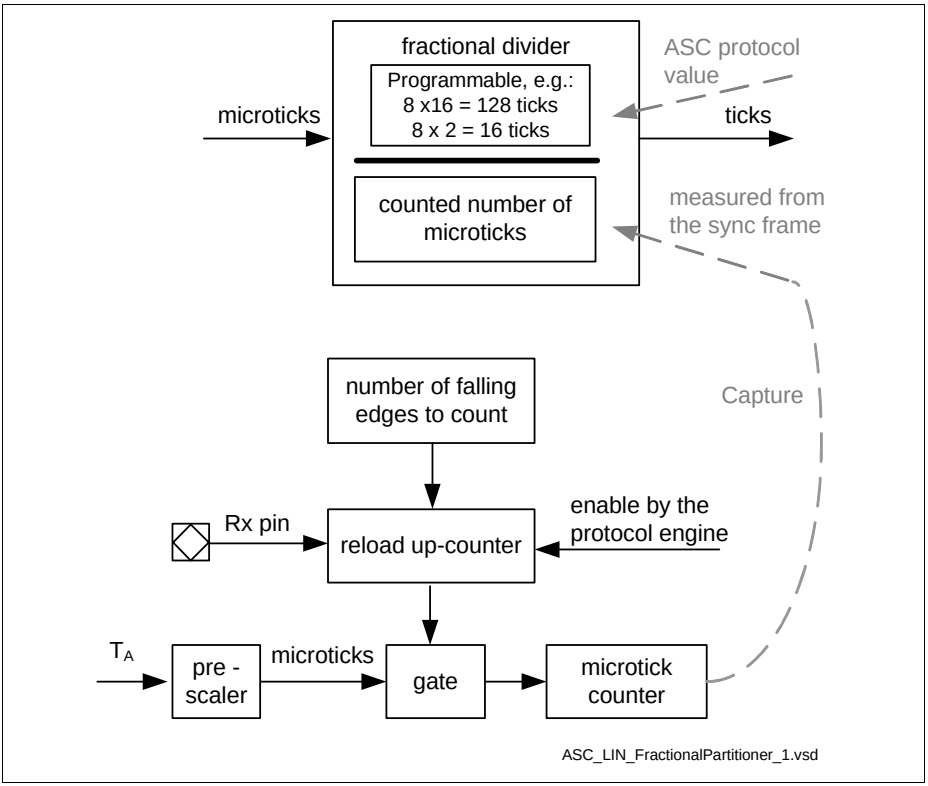


Figure 19-26 Overview of the LIN Auto Baud Detection Principle

Auto baud rate detection measures the time between the first and the fifth falling edge of the sync field in T_{SYS} units and loads this number in the denominator. The LIN protocol uses baud rates in a range between 2400 Baud to 19200 Baud.The expected time at 19.2 KBaud is 8 * 52.1 us = 416.8 us abd the expected denominaror value at T_{SYS} = 10ns is in the range of 417us / 10ns = 41 700.

The numerator operates internally with the quantums for 8 bits, 16 times oversampling for LIN protocol: 8*16=128. The bit field **BRG**.NUMERATOR is ignored.

19.11 Collision Detection

Collison detection monitors the consistency of transmitted and the echoed received bytes in LIN mode and half duplex SPI mode.

19.12 LIN Protocol Control

There is a central LIN protocol state machine. It is connected to the receiver and the transmitter shift registers, the Tx and Rx FIFOs, the checksum logic and the watchdog timers. The machine takes care of generating the sync byte of 55_H and the automatic handling of the checksum. Both the classic LIN V1.3 and the enhanced LIN V2.0 / V2.1 checksum are supported (>> **LINCON** register). The hardware checksum feature is switchable on and off, and the choice between using the classical and the enhanced checksum is done by software with the **DATCON**.CSM bit on a frame by frame basis. As can be seen in the **Figure 19-28**, for the LIN version 2.0 and 2.1, the enhanced checksum is calculated for the identifiers 0...59, and the classical checksum for the identifiers 60...63.

Additionally, the parity of the ID field is generated in master mode, and checked in slave mode. In slave mode, in case of a mismatch between the received and the calculated parity, an error interrupt is raised.

In receive case, if **LINCON**.CSI = 1, the received checksum byte is written to the RXFIFO after the last data byte.

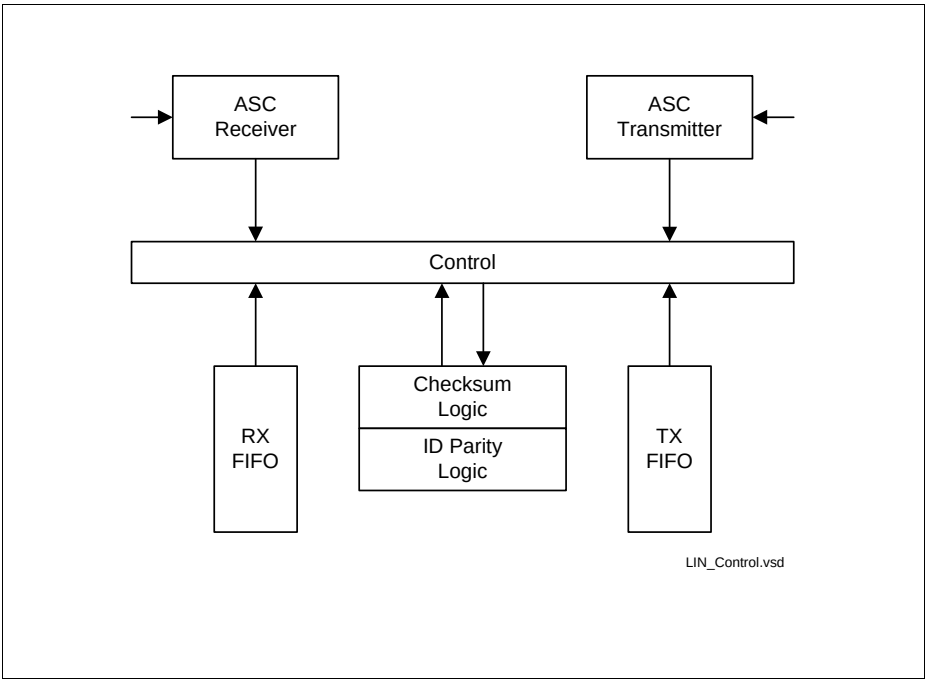


Figure 19-27 Block Diagram of the LIN Control Subsystem

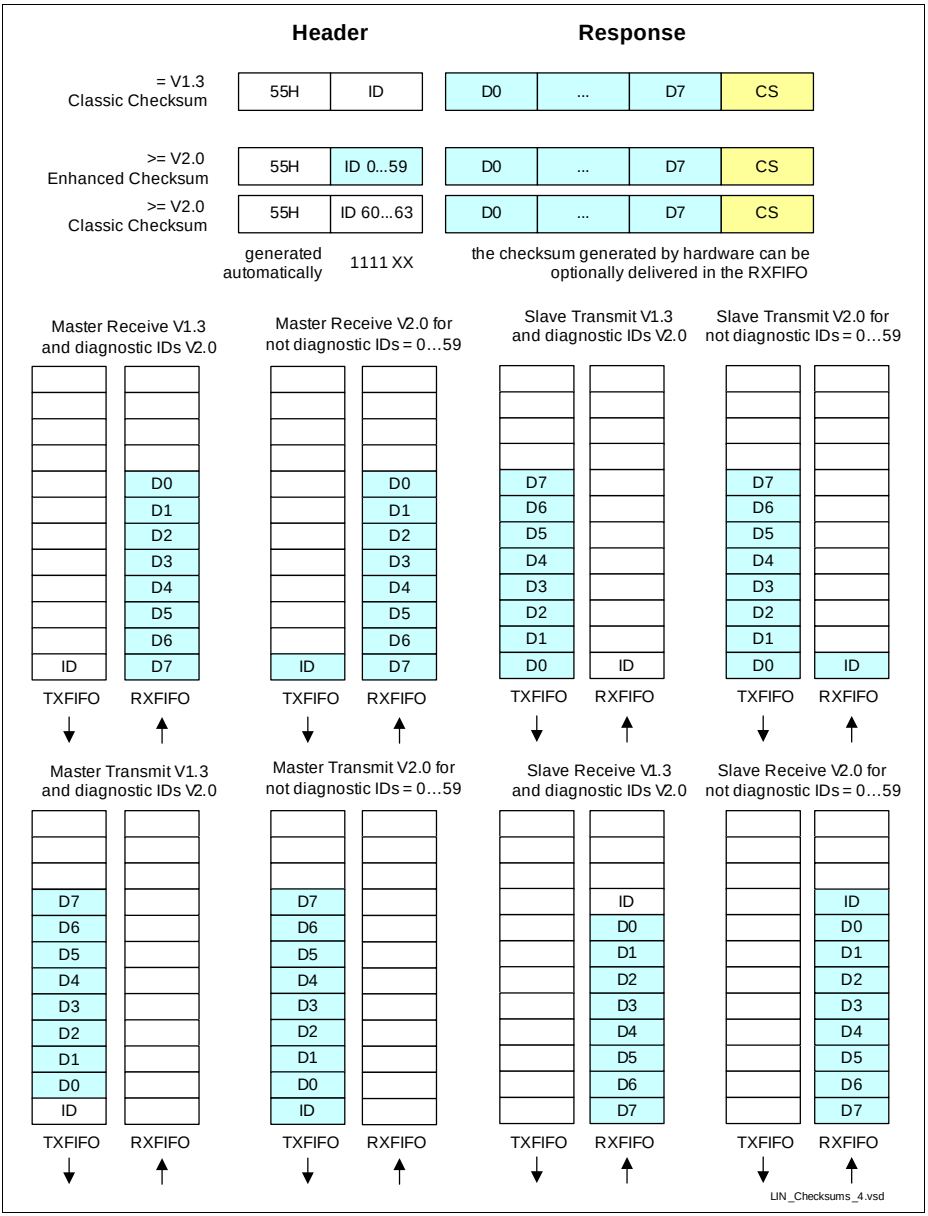


Figure 19-28 Coverage of the Checksum in Different Use Cases

19.13 Interrupts

The ASCLIN module generates three interrupts:

- TX - Transmit Interrupt
  - signals the TxFIFO level event (**FLAGS**.TFL)
- RX - Receive Interrupt
  - signals the RxFIFO level event (**FLAGS**.RFL)
- EX - Extended Error Interrupt
  - signals every error (**FLAGS**.PE, FE, CE, RFO, RFU, TFO) and
  - some additional events (**FLAGS**.FED, RED)

The LIN events related to transmission and reception of header and response are mapped to the corresponding transmit, receive and extended error interrupts

- TX:
  - transmission of the header in master mode completed (**FLAGS**.TH)
  - transmission of a response completed (**FLAGS**.TR)
- RX:
  - reception of the header in a slave mode completed (**FLAGS**.RH)
  - reception of the response completed (**FLAGS**.RR)
- EX:
  - LIN protocol (**FLAGS**. BD, TC) and
  - LIN error events (**FLAGS**.HT, RT, LP, LA, LC)

In order to determine which event is the source of a LIN interrupt, the **FLAGS** register must be polled.

Triggering a DMA

The interrupt signals are used also as DMA trigger signals. The interrupt signals are connected to the Interrupt Router Module, which routes the interrupts either to a CPU or to a DMA. There are no separate DMA triger signals.

Relationship between the Service Request Nodes and the Event Nodes

There are several events mapped to each service request node.

Each service request (interrupt) node contains an SRC (Service Request Control) register containing a sticky flag bit and the associated set, clear and enable bits. The SRC registers are located in the IR (Interrupt Router) module.

The service request flags in the SRC registers are set by hardware, and if the enable bit is set, will be cleared by hardware when the interrupt servicing starts.

One interrupt node may be triggered by more than one events. Each event which causes an interrupt also has a sticky flag bit and associated set, clear and enable bits. These bits are distributed in four registers: **FLAGS**, **FLAGSSET**, **FLAGSCLEAR**, and **FLAGSENABLE** located in the ASCLIN module. Each set of four bits associated to one event builds a virtual “event node”, see **Figure 19-29**.

The event flags are set by hardware and if enabled, trigger an interrupt. They are not cleared by hardware. They are cleared only by software in the corresponding interrupt service routine, which usually polls the flags to find the cause for the interrupts. The event flags can be also set by software for test purposes.

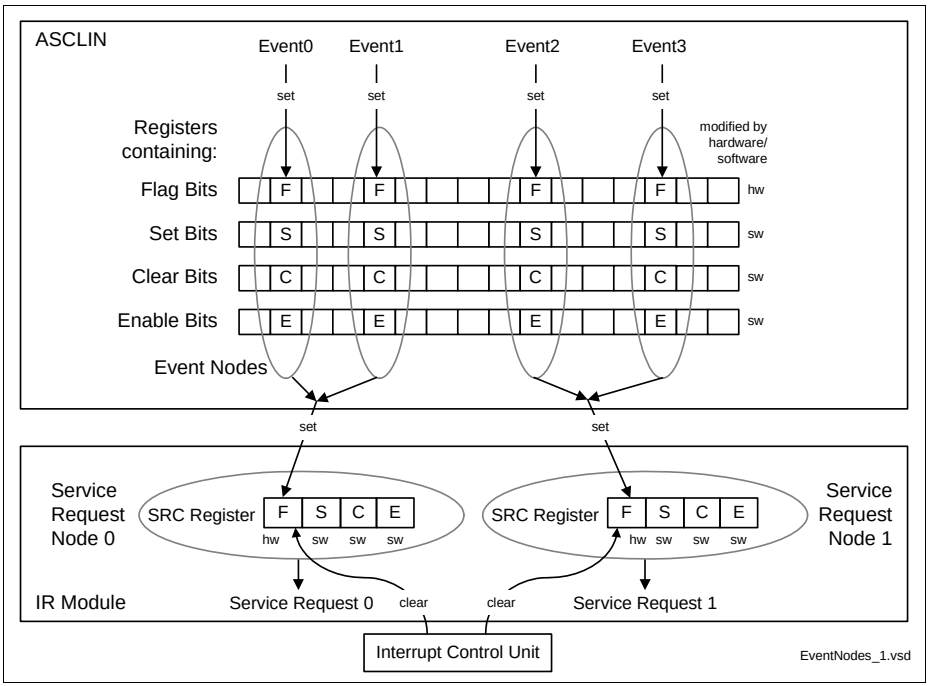


Figure 19-29 Relationship between Event Nodes and Service Request Nodes

Some flags do not generate interrupts and do not have enable bits. They are used either for issueing requests, which are subsequently acknowledged by hardware, or only for polling by software, see **Figure 19-30**.

Clearing an request / acknowledge flag by software is ignored (for example **FLAGS**.TRRQ, THRQ, TWRQ).

There are no pure polled only flags in the ASCLIN module, although by letting the enable bits disabled, all interrupt event flags can be used as polled only (for example idle and stuck monitoring by using **FLAGS**.FED, RED). Setting such bits per software in such a case is meaningless, except for test purposes.

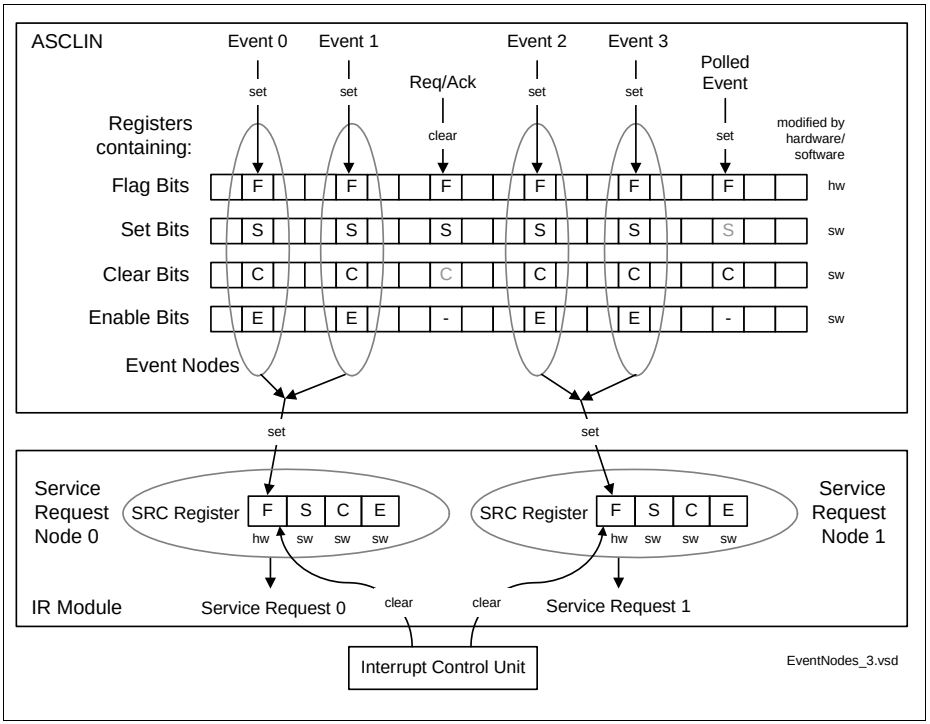


Figure 19-30 Acknowledge Flags and Polled Flags

19.14 Digital Glitch Filter

The digital glitch filter removes short glitches from the input data signal by using an increment / decrement counter with programmable threshold. On the other hand, the filter introduces a delay into the signal path, depending on the digital filter sampling frequency f_{PD} and the threshold programmed in the bit field **IOCR**.DEPTH.

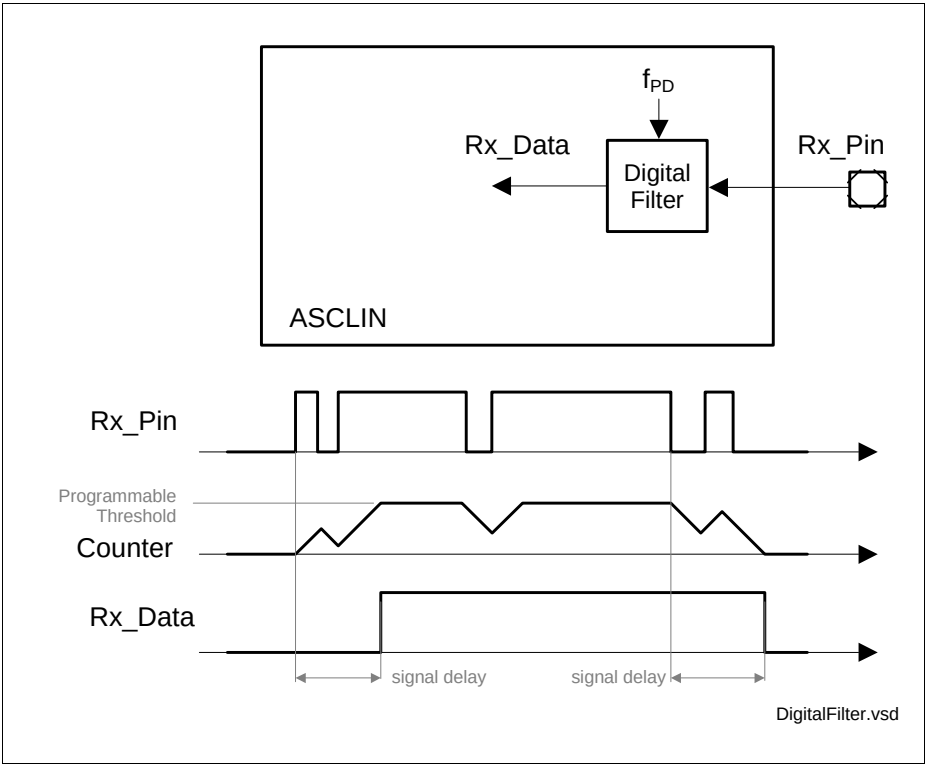


Figure 19-31 Digital Filter

19.15 Suspend, Sleep and Power-Off Behavior

Generally, the ASCLIN module transmits short data frames, not longer than 16 bits, mostly 8 bit long, sometimes with very low baud rates, down to few KBaud. In LIN mode, the data frames build LIN frames.

Simply freezing the module in the middle of an asynchronous ASC frame mostly results in an erroneous data at the receiver side, depending on the data content.

Freezing the module inside the LIN frame transmission, but between the transmission of single bytes results in a timeout error at the receiver side.

If the ASCLIN module gets a request for switching off the clock, it disables immediately all request lines towards the DMA in order to stop further DMA transfers.

In order to power down the module in a well defined way, the user software must take care that the power down request is issued when there is no ongoing single frame or LIN frame transfer and that both FIFOs are empty.

19.15.1 OCDS Suspend

OCDS soft suspend request suspends the module activity at the end of the current transaction. In ASC and SPI cases, this is the end of the current frame. In LIN case, this is the end of the current LIN frame (regular end of response, header timout, or response timeout).

OCDS hard suspend request, for debugging purposes, immediately freezes the ASCLIN module in the current state. The SPB clock feeding the kernel clock f_{CLC} is immediatly switched off. If f_{CLC} is selected for f_A (by using the **CSR**.CLKSEL bit field), the asynchronous clock f_A is also immediately switched off.

*Note: The asynchronous clock f_A remains switched on if a clock source other than f_{CLC} is selected for f_A (by using the **CSR**.CLKSEL bit field).*

Note: Reading and writing of registers is possible but will enable the kernel clock f_{CLC} for a few cycles. Attention: register accesses with clocking in Hard Suspend Mode can have unintended side effects like signals becoming and staying active. This can affect also other modules, so a ASCLIN kernel reset might not be sufficient to bring the system into a defined state.

19.15.2 Sleep Mode

Going into sleep mode implies that the ASCLIN module has finished all activities and that the TX and RX FIFOs are empty. It is the responsibility of the user software to issue a sleep request in a safe time interval, where no race conditions or pipeline effects between the ASCLIN module and the DMA can occur.

The ASCLIN module behaves in the same way as in case of disable request..

19.15.3 Disable Request (Power-Off)

Going into power off implies that the ASCLIN module has finished all activities and that the TX and RX FIFOs are empty. It is the responsibility of the user software to issue a power-off request in a safe time interval, where no race conditions or pipeline effects between the ASCLIN module and the DMA can occur. This is no issue if the power-off procedure is one-way, that is no continuation of operation without reinitialization of the module is expected.

The ASCLIN module sets immediately all outputs to inactive, stops reacting to inputs, the internal state machines go to initial state, the registers remain unchanged, and then the module acknowledges the request. All clocks to the module are switched off.

19.16 Reset Behavior

There are two sources of reset:

- BPI (Bus Peripheral Interface)
- Reset bit in the kernel register KRSTx0.RST and KRSTx1.RST.

Both sources reset the ASCLIN module. They execute an Application Reset, which resets all the registers except the OCS register. All output signals get the reset value.

The **OCS** register is reset by the Debug Reset.

19.17 Use Case Example ASC Interface

This section explains the core functionality of the ASCLIN interface with a code example. The example uses the ASC feature of the module to transmit and receive a data set via loop back mode (see also figure 21-2 (LINK!)). The following settings are used:

- Data length: 8 bit
- Baud rate: 9600 Bd
- One stop bit, parity bit checking, oversampling factor 16, sampling points 7,8 and 9
- Rx and Tx interrupt

Step description to initialize the ASC module with the settings from above:

(Line 1) reset ENDINIT to get access to ENDINIT protected register, here ASCLIN0_CLC. (see also ENDINIT protection chapter 8.9.3(LINK))

(Line 2) enable the control of the module in the clock control register CLC (LINK)

(Line 3) store CLC register in a dummy variable (has to be defined before). Read back to avoid pipeline effects.

(Line 4) set ENDINIT to lock the protected register again.

(Line 5) The clk source gets set in the clock selection register CSR (LINK). Before setting a source, no clk supply ([3:0]=0) must be set. Final clk selection in Line 20.

(Line 6) This line sets the loop back mode in the input and output control register IOCR (LINK). (see also figure 21-2 (LINK))

(Line 7) This line clears and enables the Tx FIFO and also defines the writing size of 1 byte per clk. The Tx FIFO filling level to trigger an interrupt gets set to 0 ([11:8] =0). The Tx FIFO triggers a Tx interrupt, if the Tx FIFO filling level falls to or below that defined level. The Tx interrupt get enabled in Line 14. (see also chapter 21.4.1 TxFIFO Overview (LINK) and TXFIFOCON (LINK) register)

(Line 8) This line installs the Rx FIFO identically like the Tx FIFO from the line above. (see also RXFIFOCON (LINK) and chapter 21.4.2 RxFIFO Overview (LINK))

(Line 9) The oversampling factor (here 16), the sample points (here 7,8 and 9) and the prescalar for the baudrate (here 10) gets configured in the bit configuration register BITCON (LINK). (see also section 21.5. Clock System (LINK))

(Line 10) One stop bit and initialize mode as basic operation mode (necessary before switching to another mode) are getting configured in the FRAMECON (LINK) register. The parity bit feature with parity type even gets also enabled.

(Line 11) The data length of 8 bits gets set in the DATCON (LINK) register.

(Line 12) the clk divider for the baud rate gets set to 48/3125 in the baud rate generation register BRG (LINK) (see also 21.5. Clock System (LINK))

*Note: assuming f_{clc} = 100 MHz and (defined in line 8) prescalar = 10, oversampling = 16: f_{shift} = (f_{clc}/prescalar * 48/3125) / oversampling =* ***9600 Bd***

(Line 13) Clear all interrupt flags in the FLAGSCLEAR (LINK) register. Before setting the respective interrupt flags in Line 14

(Line 14) enable the Tx and Rx interrupts in the FLAGSENABLE (LINK) register. The interrupts getting triggered by the FIFO's (see Line 6/7).

(Line 15) This line enables the Tx interrupt in the service request control register SRC_ASCLIN0TX (LINK) and sets the interrupt priority to ASC0TX_PRIO (1...255).

(Line 16) This line enables the Rx interrupt in the service request control register SRC_ASCLIN0RX (LINK) and sets the interrupt priority to ASC0RX_PRIO (1...255).

(Line 17 and 18) The interruptHandlerInstall function gets called (this function has to be written first, see interrupt handler example (LINK) for more details). This function installs the interrupt service routine (ISR) entry address in the interrupt vector array with the priority ASC0TX_PRIO respectively ASC0RX_PRIO.

(Line 19) setting operating mode finally to ASC (see also Line 9).

(Line 20) setting clk source finally to f_{clc} (see also Line 4).

Note: your Tx ISR function prototype would be: void ASC0_TX_ irq (void); here could be your ISR code;

Initialization of the ASC interface:

```
(1)  SCU_vResetENDINIT (0);     // enable ENDINIT register
(2)  ASCLIN0_CLC = 0x000;       // disable module control
(3)  dummy = ASCLIN0_CLC;       //read back
(4)  SCU_vSetENDINIT (0);       // lock ENDINIT register
(5)  ASCLIN0_CSR = 0;           // clk source, no clk
(6)  ASCLIN0_IOCR = 0x10000001;           // Loopback
(7)  ASCLIN0_TXFIFOCON = 0x00000043;      // init TXFIFO
(8)  ASCLIN0_RXFIFOCON = 0x00000043;      // init RXFIFO
(9)  ASCLIN0_BITCON = 0x830F0009;    // OS 16, SP 7,8,9, PS 10
(10) ASCLIN0_FRAMECON = 0x40000200;       // 1 Stop, Init Mode, P
(11) ASCLIN0_DATCON = 0x7;      //8 Data Bits
(12) ASCLIN0_BRG = 0x00300C35;       // divider for Baudrate
(13) ASCLIN0_FLAGSCLEAR = 0xFFFFFFFFF;    // Clear all Flags
(14) ASCLIN0_FLAGSENABLE = 0xF0000000;    // Enable TX/RX Int.

(15) SRC_ASCLIN0TX = (1 << 10) | ASC0TX_PRIO;

(16) SRC_ASCLIN0RX = (1 << 10) | ASC0RX_PRIO;
```

```
(17) interruptHandlerInstall (ASC0TX_PRIO, & ASC0_TX_irq);
(18) interruptHandlerInstall (ASC0RX_PRIO, & ASC0_RX_irq);

(19) ASCLIN0_FRAMECON |= 0x00010000;// Mode ASC
(20) ASCLIN0_CSR = 1;            //clk source CLC
```

Note: Before an interrupt is able to occur, the interrupt system has to be globally enabled. The Interrupt Control Register (ICR) holds the global interrupt enable bit (ICR.IE) which enables the CPU service request system. Most compiler support the attribute (or similar):
__enable();
to set this bit. (See also Architecture Manual for more details)

19.18 Kernel Registers

This section describes the kernel registers of the ASCLIN module. All ASCLIN kernel register names described in this section will be referenced in other parts of the TC21x/TC22x/TC23x User's Manual by the module name prefix "ASCLIN0_" for the ASCLIN0 interface and "ASCLIN1_" for the ASCLIN1 interface.

All registers in the ASCLIN address spaces are reset with the application reset (definition see SCU section "Reset Operation").

ASCLIN Kernel Register Overview

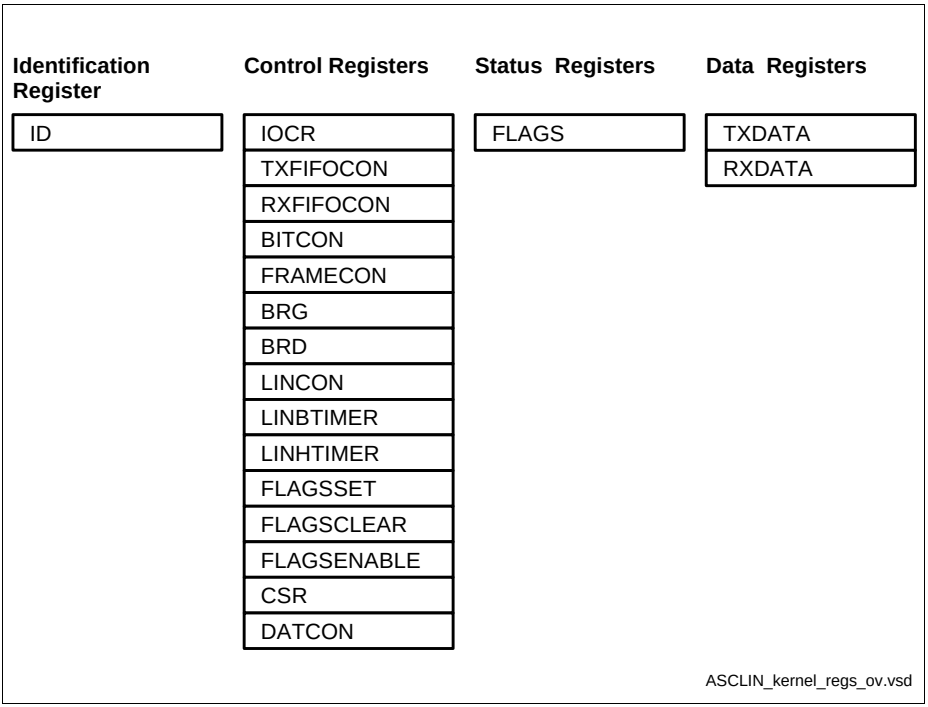


Figure 19-32 ASCLIN Kernel Registers

*Note: Absolute Register Address = Module Base Address (**Table 19-3**) + Offset Address (**Table 19-4**).*

Table 19-3 Registers Address Space

Module	Base Address	End Address	Note
ASCLIN0	$F0000600_H$	$F00006FF_H$	–
ASCLIN1	$F0000700_H$	$F00007FF_H$	–

Table 19-4 Registers Overview

Register Short Name	Register Long Name	Offset Address	Write [1] Access	Reset Value
IOCR	Input Output Control Register [2]	04_H	U, SV,P	X000 0000
ID	Module Identification Register	08_H	BE	00C1 C0XX
TXFIFOCON	TX FIFO Configuration Register	$0C_H$	U, SV,P	0000 0000
RXFIFOCON	RX FIFO Configuration Register	10_H	U, SV,P	0000 0000
BITCON	Bit Timing Configuration Reg. [2]	14_H	U, SV,P	0000 0000
FRAMECON	Frame Configuration Register [2]	18_H	U, SV,P	0000 0000
DATCON	Data Configuration Register	$1C_H$	U, SV,P	0000 0000
BRG	Baud Rate Generation Register [2]	20_H	U, SV,P	0000 0000
BRD	Baud Rate Detection Register [2]	24_H	U, SV,P	0000 0000
LINCON	LIN Configuration Register [2]	28_H	U, SV,P	0000 0000
LINBTIMER	LIN Break Timer Register [2]	$2C_H$	U, SV,P	0000 0000
LINHTIMER	LIN Header Timer Register [2]	30_H	U, SV,P	0000 0000
FLAGS	Flags Register	34_H	BE	0000 0000
FLAGSSET	Flags Set Register	38_H	U, SV,P	0000 0000
FLAGSCLEAR	Flags Clear Register	$3C_H$	U, SV,P	0000 0000
FLAGSENABLE	Flags Enable Register	40_H	U, SV,P	0000 0000
TXDATA	Transmit Data Register	44_H	U, SV,P	0000 0000
RXDATA	Receive Data Register	48_H	BE	0000 0000
CSR	Clock Selection Register	$4C_H$	U, SV,P	0000 0000
RXDATAD	Receive Data Debug Register	50_H	BE	0000 0000
BPI Registers				
CLC	Clock Control Register	00_H	SV, E, P	0000 0003
OCS	OCDS Control and Status Reg.	$E8_H$	SV, P	0000 0000
KRSTCLR	Reset Status Clear Register	EC_H	SV, E, P	0000 0000
KRST1	Reset Control Register 1	$F0_H$	SV, E, P	0000 0000
KRST0	Reset Control Register 0	$F4_H$	SV, E, P	0000 0000
ACCEN1	Access Enable Register 1	$F8_H$	SV, SE	0000 0000
ACCEN0	Access Enable Register 0	FC_H	SV, SE	FFFF FFFF

1) All registers have read access mode U, SV. Read access from reserved addresses delivers bus error (BE). All registers belong to application (class 3) reset except OCS, which belongs to debug (class 1) reset.
2) This register is writable only if **CSR**.CLKSEL=0.

List of Access Protection Abbreviations

U - User Mode

SV - Supervisor Mode

BE - Bus Error

nBE - no Bus Error

P - Access Protection, as defined by the ACCEN Register

E - ENDINIT

SE - Safety ENDINIT

19.18.1 Kernel Registers

IOCR

The Input and Output Control Register IOCRx determines several properties of the RX and TX signal path:

for the RX signal:

- the alternate input
- the filter depth

for the TX signal

- the trigger source

(>> **Table 19-4** register overview)

(>> **Baud Rate Generation**)

(>> **Bit Timing Properties**)

IOCR

Input and Output Control Register (04) **Reset Value: X000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXM	**RXM**	**CTS EN**	**LB**	**SPO L**	**CPO L**	**RC POL**	**0**							**CTS**	
rh	rh	rw	rw	rw	rw	rw	r							rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						**DEPTH**						**0**		**ALTI**	
r						rw						r		rw	

Field	Bits	Type	Description
ALTI	[2:0]	rw	**Alternate Input Select** Selects the alternate input for the RX signal: 000_B Alternate Input 0 selected 001_B Alternate Input 1 selected $\cdots_B$ … 111_B Alternate Input 7 selected

Field	Bits	Type	Description
DEPTH	[9:4]	rw	**Digital Glitch Filter Depth** DEPTH determines the number of port input samples clocked with microticks that are taken into account for the calculation of the floating average. The higher the DEPTH is chosen to be, the longer the glitches that are suppressed and the longer the delay of the input signal introduced by this filter. 000000_B off, default 000001_B 1 000010_B 2 000011_B 3 $\ldots_B$ … 111111_B 63
CTS	[17:16]	rw	**CTS Select** Selects the CTS input pin out of maximum four possible. 00_B 0 01_B 1 10_B 2 11_B 3
RCPOL	25	rw	**RTS CTS Polarity** RCPOL defines the active level or the RTS and CTS signals. Active means ready/clear to send. 0_B active high 1_B active low
CPOL	26	rw	**Clock Polarity in Synchronous Mode** CPOL defines the idle level of the clock signal if the module is set in the SPI mode. The idle level is the level outside the data transmission time intervals. Default is low level. 0_B Idle low 1_B Idle high
SPOL	27	rw	**Slave Polarity in Synchronous Mode** Defines the idle level of the SLSO signal, which is the level outside the data transmission, leading and trailing time intervals. 0_B Idle low 1_B Idle high

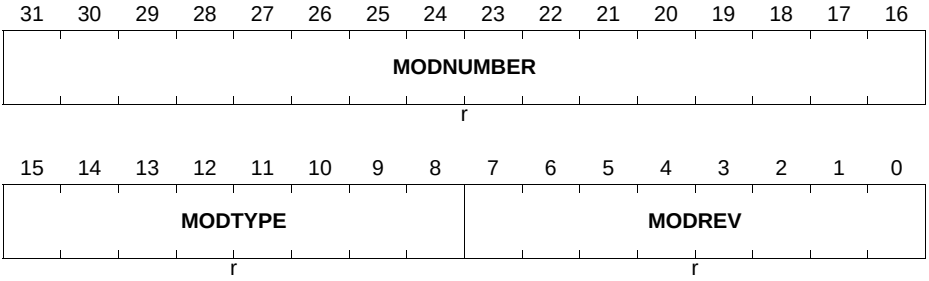
Field	Bits	Type	Description
LB	28	rw	**Loop Back Mode** Enables the in module connection of the transmit signal to receive signal. If Loop-back is enabled, the module can be run and tested without an external connection, in ASC and SPI modes. In LIN mode, loopback should not be used, because the module can be either master or slave. 0_B Disabled 1_B Enabled
CTSEN	29	rw	**Input Signal CTS Enable** Enables the sensitivity of the module to the external CTS signal. If disabled, the CTS signal is considered being permanently active. 0_B Disabled 1_B Enabled
RXM	30	rh	**Receive Monitor** Shows the status of the receive signal. 0_B Current signal is low. 1_B Current signal is high.
TXM	31	rh	**Transmit Monitor** Shows the status of the transmit signal. 0_B Current signal is low. 1_B Current signal is high.
0	3, [15:10], [24:18]	r	**Reserved** Read as 0; should be written with 0.

ID

The Module Identification Register ID contains read-only information about the module version.

(>> **Table 19-4** register overview)

ID
Module Identification Register (08$_H$) Reset Value: 00C1 C0XX$_H$



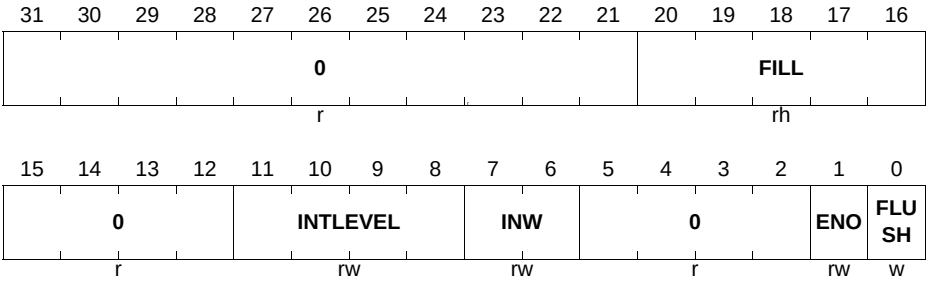
Field	Bits	Type	Description
MODREV	[7:0]	r	**Module Revision Number** MODREV defines the module revision number. The value of a module revision starts with 01$_H$ (first revision).
MODTYPE	[15:8]	r	**Module Type** This bit field is C0$_H$. It defines a 32-bit module.
MOD NUMBER	[31:16]	r	**Module Number Value** This bit field together with MODTYPE uniquely identifies a module.

TXFIFOCON

(>> **Table 19-4** register overview)

TXFIFOCON

TX FIFO Configuration Register (0C$_H$) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
FLUSH	0	w	**Flush the transmit FIFO** Write of 1 brings the Tx FIFO in empty state. Write of 0 has no effect. 0_B No effect 1_B Empty the Tx FIFO
ENO	1	rw	**Transmit FIFO Outlet Enable** Enables the TxFIFO outlet. In SPI and ASC modes, data transmission starts immediately when the data is available, whereas in LIN case the transmission start is controlled by the protocol engine. 0_B Disabled. In LIN case, if the protocol engine tries to fetch data. 1_B Enabled. In LIN case, no data is moved to the shift register until it is fetched by the protocol engine.
INW	[7:6]	rw	**Transmit FIFO Inlet Width** Defines the number of bytes written to the Tx FIFO with one FPI bus write. 00_B 0 01_B 1 10_B 2 11_B 4

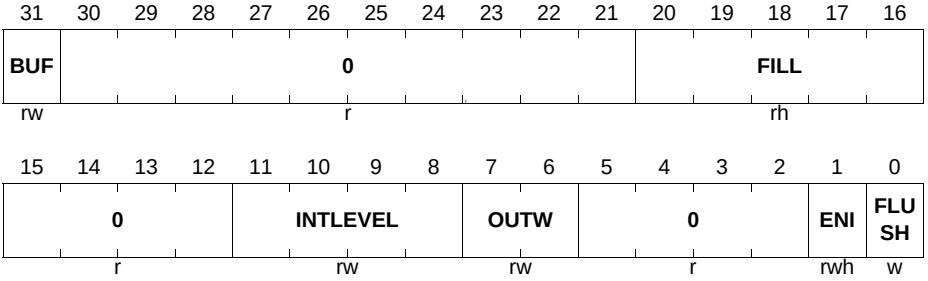
Field	Bits	Type	Description
INTLEVEL	[11:8]	rw	**FIFO Interrupt Level** Defines the filling level that triggers an re-fill interrupt or DMA access. An interrupt is generated when the filling level falls to INTLEVEL or below, each time when a data byte is taken out of the FIFO 0000_B 0 0001_B 1 0010_B 2 $...B$ ... 1111_B 15
FILL	[20:16]	rh	**FIFO Filling Level** Read only bit-field containing the current filling level of the FIFO. 00000_B 0 00001_B 1 $...B$ ... 10000_B 16 10001_B reserved 10010_B reserved 10011_B reserved 10100_B reserved 10101_B reserved 10110_B reserved 10111_B reserved 11000_B reserved 11001_B reserved 11010_B reserved 11011_B reserved 11100_B reserved 11101_B reserved 11110_B reserved 11111_B reserved
0	[5:2], [15:12], [31:21]	r	**Reserved**

RXFIFOCON

(>> **Table 19-4** register overview)

RXFIFOCON

RX FIFO Configuration Register (10_H) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
FLUSH	0	w	**Flush the receive FIFO** Write of 1 brings the Rx FIFO in empty state. Write of 0 has no effect. 0_B No effect 1_B Empty the Rx FIFO
ENI	1	rwh	**Receive FIFO Inlet Enable** Enables the receiver and the filling of the Rx FIFO through the shift register. In LIN slave mode, this bit is set by hardware after the correct reception of the sync byte. The software can clear this bit after reception of an foreign ID in order to suppress the reception of the following response. 0_B Disabled 1_B Enabled
OUTW	[7:6]	rw	**Receive FIFO Outlet Width** Defines the number of bytes read to the Rx FIFO with one FPI bus read. 00_B 0 01_B 1 10_B 2 11_B 4

Field	Bits	Type	Description
INTLEVEL	[11:8]	rw	**FIFO Interrupt Level** Defines the filling level that triggers a drain interrupt or DMA access. An interrupt is generated when the filling level rises to INTLEVEL or beyond, each time when a data byte is delivered to the FIFO. 0000_B 1 0001_B 2 $...B$ ... 1111_B 16
FILL	[20:16]	rh	**FIFO Filling Level** Read only bit-field containing the current filling level of the FIFO. 00000_B 0 00001_B 1 $...B$ ... 10000_B 16 10001_B reserved 10010_B reserved 10011_B reserved 10100_B reserved 10101_B reserved 10110_B reserved 10111_B reserved 11000_B reserved 11001_B reserved 11010_B reserved 11011_B reserved 11100_B reserved 11101_B reserved 11110_B reserved 11111_B reserved

Field	Bits	Type	Description
BUF	31	rw	**Receive Buffer Mode** If this bit is zero, then the RXFIFO behaves normally as described in the ITS If this bit is set, the RXFIFO behaves as simple 32-bit one stage RX buffer, which is overwritten with each new received data. The received bits appear in the RXDATA register on the lowest bit locations. The upper locations are padded with zeros. 0_B RXFIFO 1_B Single Stage RX Buffer
0	[5:2], [15:12], [30:21]	r	**Reserved**

BITCON

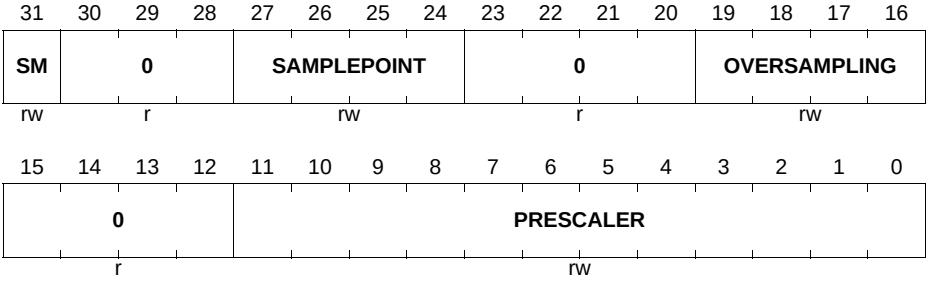
The BITCON Register defines the integer timer parameters in the baud rate generation block.

(>> **Table 19-4** register overview)

(>> **Baud Rate Generation**)

(>> **Bit Timing Properties**)

BITCON
Bit Configuration Register **(14_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
PRE SCALER	[11:0]	rw	**Prescaling of the Fractional Divider** Prescaler in the range of 1 to 4096. Used also as a microtick generator for the input digital filter.
OVER SAMPLING	[19:16]	rw	**Oversampling Factor** Defines the bit length in ticks in the range of 1 to 16. The lengths of 1 to 3 are not allowed. The position of the sampling points is shown in **Figure 19-16**. 0000_B 1 (not allowed) 0001_B 2 (not allowed) 0010_B 3 (not allowed) 0011_B 4 $\cdots_B$ ... 1111_B 16

Field	Bits	Type	Description
SAMPLE POINT	[27:24]	rw	**Sample Point Position** Programmable in the range of 0 to 15 according to the Figure 19-16. For example, if three sample points at position 7, 8, 9 are required, this bit field would contain 9. 0000_B 0 (not allowed) 0001_B 1 $...{}_B$ ... 1111_B 15 In SPI mode, this bit field + 1 defines the length of the first SCLK half period in ticks. Values equal or higher then the OVERSAMPLING value are forbidden.
SM	31	rw	**Sample Mode** Number of samples per bit. 0_B 1 1_B 3
0	[15:12], [23:20], [30:28]	r	**Reserved**

FRAMECON

The parameters regarding the properties of the message frame of the ASC communication are controlled by the Frame Control Register FRAMECON.

(>> **Table 19-4** register overview)

FRAMECON
Frame Control Register **(18_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ODD	**PEN**	**CEN**	**MSB**	**0**										**MODE**	
rw	rw	rw	rw											rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**LEAD**			**STOP**			**IDLE**			**0**					
r	rw			rw			rw			r					

Field	Bits	Type	Description
IDLE	[8:6]	rw	**Duration of the IDLE delay** Defines the duration of the IDLE delay in bit times. If more characters are available in the TXFIFO, this is the pause inserted between the characters. In the SPI mode, this is the idle time between the frames. In the ASC and LIN mode, this is the pause inserted between transmission of bytes. Idle also applies to the pause between the header and the response (response space). *Note: The collision detection runs in parallel to the idle delay and in LIN mode may extend the time between two bytes for one bit length. This effect may occure if the the round trip delay including the digital filter delay is longer than the idle delay.* 000_B 0 001_B 1 ...B ... 111_B 7
STOP	[11:9]	rw	**Number of Stop Bits** Defines the number of stop bits in ASC and LIN mode, or the trailing delay in SPI mode. In ASC mode, standard values are 1 and 2. In LIN mode, standard value is 1. In SPI mode there is no standard value. Nevertheless, all settings are possible in all modes. 000_B 0 (not allowed in ASC and LIN modes) 001_B 1 ...B ... 111_B 7
LEAD	[14:12]	rw	**Duration of the Leading Delay** Defines the leading delay in bit times in SPI mode. Has no meaning in the ASC mode. In LIN mode, this is a delay inserted between the end of the break and the start of the sync character. 000_B 0 (not allowed in LIN mode) 001_B 1 ...B ... 111_B 7

Field	Bits	Type	Description
MODE	[17:16]	rw	**Mode Selection** This bit field defines the basic operating mode of the module. In initialize mode, all outputs are at inactive level, an the module does not respond to the input signals. Changing the mode of the module must be done by switching first to initialize mode, and then to the other mode. The SCLK signal generated by the module is active only in the SPI mode. The CTS output generated by the module is active only in the ASC mode. 00_B Initialize mode 01_B ASC mode 10_B SPI mode 11_B LIN mode
MSB	28	rw	**Shift Direction** Defines the shift direction of the shift register. Relevant for the SPI mode. In ASC and LIN modes, should be set to zero. Parity bit is shifted out last independently of the shift direction. 0_B LSB first 1_B MSB first
CEN	29	rw	**Collision Detection Enable** Enables the collision detection mechanism. 0_B Disabled 1_B Enabled
PEN	30	rw	**Parity Enable** Enables the parity bit attached to the data bits. Parity bit can be used for ASC and SPI protocols. The standard LIN bytes do not use this parity bit. 0_B Disabled 1_B Enabled

Field	Bits	Type	Description
ODD	31	rw	**Parity Type** Defines the type of parity bit attached to the data bits. This setting is valid for all modes of operation (ASC, LIN, SPI). 0_B Even 1_B Odd
0	[5:0], 15, [27:18]	r	**Reserved** Read as 0; should be written with 0.

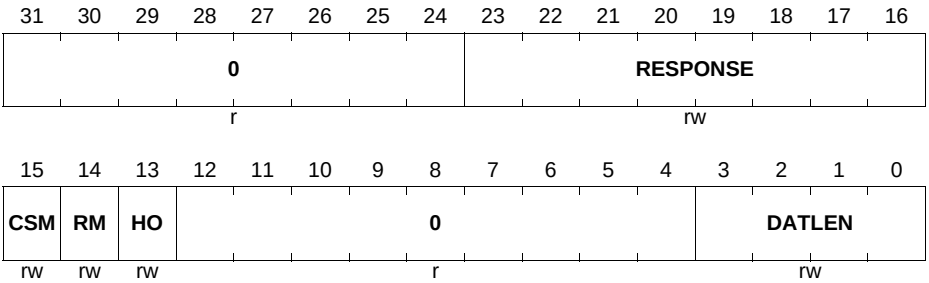
DATCON

DATCON register defines the number of bits in the ASC (affecting the LIN protocol, if used) and SPI frames, data length or the number of data bytes in a LIN response (if any) and the checksum mode. It additionly defines the time window for the LIN response.

In case the whole LIN response does not fit in this time window, an error interrupt is generated. The measurement starts at the end of the last bit of the header, and ends at the end of the last bit of the response.

(>> **Table 19-4**)

DATCON
Data Configuration Register **($1C_H$)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RESPONSE	[23:16]	rw	**Response Timeout Threshold Value** Defines the timer limit in the range of 1 to 256 bit times.
HO	13	rw	**Header Only** Defines if the LIN frame shall consist of a header and response or of a header only. 0_B Header and response expected 1_B Header only expected, response ignored
RM	14	rw	**Response Mode** Defines if the RESPONSE bit field defines a LIN Response or LIN Frame timeout threshold. See **Figure 19-23**. 0_B Frame 1_B Response

Field	Bits	Type	Description
CSM	15	rw	**Checksum Mode** Defines if the classic or the enhanced checksum will be calculated by the checksum block. 0_B Classic 1_B Enhanced
DATLEN	[3:0]	rw	**Data Length** Defines the number of bits in a character. In the ASC mode, standard length is 7, 8, or 9 bits. In the SPI mode, there is no standard length. In ASC and SPI modes, any length from 2 to 16 bits is possible, although not standard for some protocols. In LIN mode, standard length is 8 bits per character. Therefore, this field defines the number of data bytes of the response. 0000_B 1 0001_B 2 0010_B 3 0011_B 4 $...{}_B$ ... 1111_B 16
0	[31:24], [12:4]	r	**Reserved**

BRG

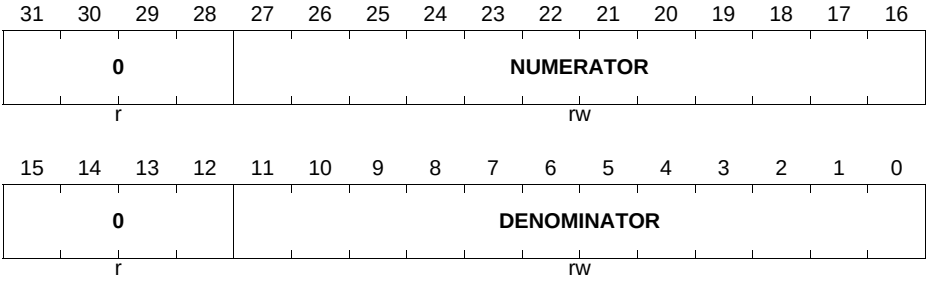
Configures the numerator and the denominator of the fractional divider in the baud rate generation block.

(>> **Table 19-4** register overview)

(>> **Baud Rate Generation**)

BRG

Baud Rate Generation Register (20_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
NUMERATOR	[27:16]	rw	**Numerator** Defines the numerator of the fractional divider in a range of 0 to 4095. Programmed by software. The setting of 0 is not allowed.
DENOMINATOR	[11:0]	rw	**Denominator** Programmed by software, in a range of 0 to 4095. The setting of 0 is not allowed If the module is used as ASC, SPI, LIN master and LIN slave without autobaud detection, this value determines the baud rate. In slave mode with autobaud detection, it contains the nominal value. For the value measured by the autobaud detection hardware, see the **BRD** register.
0	[15:12], [31:28]	r	**Reserved**

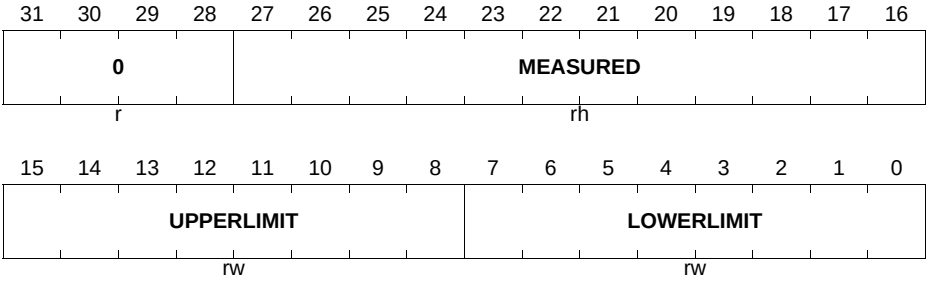
BRD

The BRD defines the properties specific for the automatic baud rate detection when the module operates as a LIN slave.

(>> **Table 19-4** register overview)

(>> **Auto Baud Rate Detection**)

BRD
Baud Rate Detection Register **(24_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
LOWER LIMIT	[7:0]	rw	**Lower Limit** This field defines the 8 most significant bits of the 12 bit compare value. The lower four bits are 1000_B. See **Auto Baud Rate Detection**.
UPPER LIMIT	[15:8]	rw	**Upper Limit** This field defines the 8 most significant bits of the 12 bit compare value. The lower four bits are 1000_B. See **Auto Baud Rate Detection**.
MEASURED	[27:16]	rh	**Measured Value of the Denominator** This bit field contains the measured value of the duration of 8-bits form the sync field of the LIN header in microtics. It is automatically loaded in the denominator of the fractional divider, in case of LIN slave operation with autobaud detection.
0	[31:28]	r	**Reserved**

LINCON

LINCON contains bits that control LIN specific features of the module.

(>> **Table 19-4** register overview)

(>> **LIN Protocol Control** chapter)

(>> **LIN Master Sequences**, **LIN Slave Sequences**)

LINCON
LIN Control Register **(28_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				ABD	MS	CSEN	0	CSI	0						
r				rw	rw	rw	r	rw	r						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
CSI	23	rw	**Checksum Injection** Defines if the received checksum byte is written into the RXFIFO or not. See **LIN Protocol Control**. 0_B Not written 1_B Written
CSEN	25	rw	**Hardware Checksum Enable** Enables the hardware checksum generation and checking. 0_B Disabled 1_B Enabled
MS	26	rw	**Master Slave Mode** Configures if the module in LIN mode operates as master or as slave. 0_B Slave 1_B Master

Field	Bits	Type	Description
ABD	27	rw	**Autobaud Detection** Enables the autobaud detection feature in LIN slave mode. In all other operating modes of the module (LIN master, ASC, SPI) not effective. If the autobaud detection is disabled (the oscillator precision of the slave is sufficient), the measurement is still active and the **FLAGS**.LA will be set when the value is outside the **BRD**.LOWERLIMIT and .UPPERLIMIT range. Additionally the stop bit of the sync field is checked if correct. If not correct, a framing error is triggered. 0_B Disabled 1_B Enabled
0	24, [22:0], [31:28]	r	**Reserved** Read as 0; should be written with 0.

LINBTIMER

This register defines

- break detection limit if the module operates as LIN slave or
- length of the generated break pulse if the module operates as LIN master.

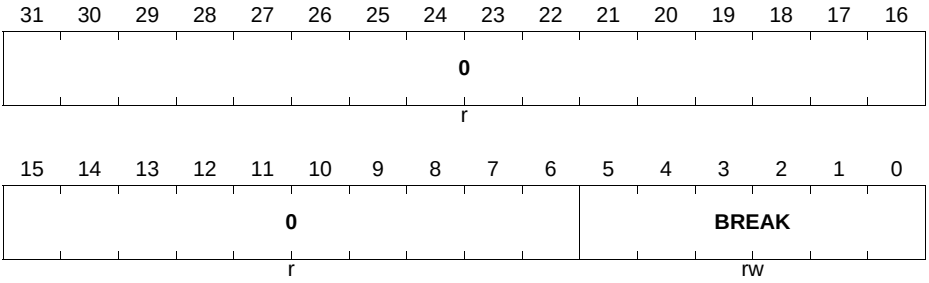
The break timer, if enabled, monitors the bus continuously.

(>> **Table 19-4** register overview)

(>> **LIN Break, Wake, Stuck Handling**)

(>> **LIN Master Sequences**, **LIN Slave Sequences**)

LINBTIMER
LIN Break Timer Register **($2C_H$)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
BREAK	[5:0]	rw	**Break Pulse Generation and Detection** In LIN slave mode, this bit field defines the duration of the detection threshold for the break pulse. In LIN master mode, this bit field defines the duration of the transmitted break pulse. The time unit is bit time.
0	[31:6]	r	**Reserved** Read as 0; should be written with 0.

LINHTIMER

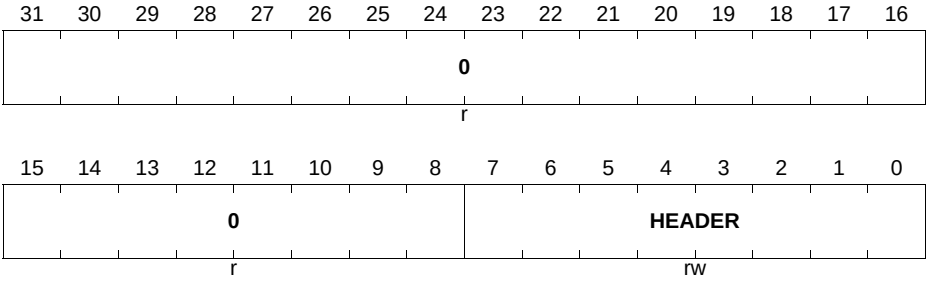
LINHTIMER register defines the time windows for the header of a LIN frame.

In master mode, the timer starts counting at the falling edge of the break pulse, and stops counting when the last bit of the header (including the stop bits) has been transmitted. If the predefined time in the HEADER bit field is violated, an error interrupt is generated (if enabled).

In slave mode, the timer starts counting when the break pulse has been detected, after a low time of 10 or 11 bit times, and stops counting when the last bit of the header has been received. If the predefined time in the HEADER bit field is violated, an error interrupt is generated (if enabled).

(>> **Table 19-4** register overview)

LINHTIMER
LIN Header Timer Register **(30_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
HEADER	[7:0]	rw	**Header Timeout Threshold Value** Defines the timer limit in the range of 1 to 256 bit times.
0	[31:8]	r	**Reserved** Read as 0; should be written with 0.

FLAGS

The FLAGS register contains all the flag bits of the ASCLIN module: the LIN phase flags (header and response transmit and receive), the overflow flags of the LIN timers, and the standard ASC error flags.

A corresponding interrupt triggering can be enabled using the **FLAGSENABLE** register.

(>> **Table 19-4** register overview)

(>> **LIN Master Sequences**, **LIN Slave Sequences**)

FLAGS
Flags Register **(34_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TFL	**TFO**	**0**	**RFL**	**RFU**	**RFO**	**CE**	**LC**	**LA**	**LP**	**BD**	**RT**	**HT**	**FE**	**TC**	**PE**
rh	rh	r	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRRQ	**THRQ**	**TWRQ**	**0**						**RED**	**FED**	**0**	**RR**	**RH**	**TR**	**TH**
rh	rh	rh	r						rh	rh	r	rh	rh	rh	rh

Field	Bits	Type	Description
TH	0	rh	**Transmit Header End Flag** Signals the HEADER_TX_END event. Set by hardware, clear by software. If enabled, a transmit interrupt is triggered. 0_B No HEADER_TX_END event since the last clear by software 1_B New HEADER_TX_END event since the last clear by software
TR	1	rh	**Transmit Response End Flag** Signals that RESPONSE_TX_END event. Set by hardware, clear by software. If enabled, a transmit interrupt is triggered. 0_B No RESPONSE_TX_END event since the last clear by software 1_B New RESPONSE_TX_END event since the last clear by software

Field	Bits	Type	Description
RH	2	rh	**Receive Header End Flag** Signals that HEADER_RX_END event. Set by hardware, clear by software. If enabled, a receive interrupt is triggered. 0_B No HEADER_RX_END event since the last clear by software 1_B New HEADER_RX_END event since the last clear by software
RR	3	rh	**Receive Response End Flag** Signals that RESPONSE_RX_END event. Set by hardware, clear by software. If enabled, a receive interrupt is triggered. 0_B No RESPONSE_RX_END event since the last clear by software 1_B New RX_RESPOSE_END event since the last clear by software
FED	5	rh	**Falling Edge from Level 1 to Level 0 Detected** This bit is set by hardware when a falling edge is detected on the RX line. 0_B No falling edge detected 1_B Faling edge detected
RED	6	rh	**Rising Edge from Level 0 to Level 1 Detected** This bit is set by hardware when a rising edge is detected on the RX line. 0_B No rising edge detected 1_B Rising edge detected
TWRQ	13	rh	**Transmit Wake Request Flag** Signals that transmission of wake has been requested. No interrupt triggered. As soon as the wake pulse transmission starts, the bit is cleared by the hardware. 0_B No Transmit Wake Request pending 1_B Transmit Wake Request pending.
THRQ	14	rh	**Transmit Header Request Flag** Signals that transmission of header has been requested. No interrupt triggered. As soon as the header transmission starts, the bit is cleared by the hardware. 0_B No Transmit Header Request pending 1_B Transmit Header Request pending.

Field	Bits	Type	Description
TRRQ	15	rh	**Transmit Response Request Flag** Signals that transmission of response has been requested. No interrupt triggered. As soon as the response transmission starts, the bit is cleared by the hardware. 0_B No Transmit Response Request pending 1_B Transmit Response Request pending.
PE	16	rh	**Parity Error Flag** Signals parity error. If enabled, an error interrupt is triggered. Parity error occurs if the internally calculated parity bit is not equal to the received parity bit. 0_B Last message received error free 1_B Last message received with parity error
TC	17	rh	**Transmission Completed Flag** Signals an end of an ASC or SSC frame. This bit is set after the last stop bit transmission in ASC mode, or after the trailing delay in case of SPI mode. If enabled, an EX interrupt is triggered. Should be cleared by software. 0_B No end of frame event pending 1_B End of frame event pending
FE	18	rh	**Framing Error Flag** Signals framing error. If enabled, an error interrupt is triggered. Framing error occurs if “0” is received at a stop bit position. If autobaud detection is deactivated, then the sync field is checked for framing error. 0_B Last message received error free 1_B Last message received with error
HT	19	rh	**Header Timeout Flag** Signals violation of the header duration limit. If enabled, an error interrupt is triggered. 0_B No HEADER_OVERFLOW event since the last clear by software 1_B New HEADER_OVERFLOW event since the last clear by software

Field	Bits	Type	Description
RT	20	rh	**Response Timeout Flag** Signals violation of the response or frame duration limit as defined in **DATCON**.RM bit. If enabled, an error interrupt is triggered. 0_B No timout event since the last clear by software 1_B New timout event since the last clear by software
BD	21	rh	**Break Detected Flag** Signals a detection of a break pulse. If enabled, an error interrupt is triggered. Slave mode only. 0_B No BWS_OVERFLOW event since the last clear by software 1_B New BWS_OVERFLOW event since the last clear by software
LP	22	rh	**LIN Parity Error Flag** Signals parity error in the LIN identifier. If enabled, an error interrupt is triggered. Applies to LIN mode only. LIN parity error occurs if the internally calculated parity bits are not equal to the received parity bits. 0_B Last ID received error free 1_B Last ID received with parity error
LA	23	rh	**LIN Autobaud Detection Error Flag** Signals baudrate outside the range defined by **BRD**.LOWERLIMIT and **BRD**.UPPERLIMIT. 0_B No autobaud detection error 1_B Autobaud detection error
LC	24	rh	**LIN Checksum Error Flag** Signals checksum error when receiving response, if the internally calculated checksum is different than the received checksum. If enabled, an error interrupt is triggered. 0_B Last checksum error free 1_B Last checksum shows an error

Field	Bits	Type	Description
CE	25	rh	**Collision Detection Error Flag** When transmitting, signals if the transmitted data differs from the received data. If enabled, an error interrupt is triggered in case of a mismatch. Collision detection is mandatory only when supporting LIN version 2.1. 0_B No mismatch 1_B Mismatch detected
RFO	26	rh	**Receive FIFO Overflow Flag** Signals an overflow error. If enabled, an error interrupt is triggered. 0_B No overflow error pending 1_B Overflow error pending
RFU	27	rh	**Receive FIFO Underflow Flag** Signals an underflow error. If enabled, an error interrupt is triggered. See also **RxFIFO Overview**. 0_B No underflow error pending 1_B Underflow error pending
RFL	28	rh	**Receive FIFO Level Flag** An interrupt (if enabled) is generated when the RXFIFO filling level rises to INTLEVEL or above, each time when a data byte is delivered to the RXFIFO. This flag signals that the event from above occured. 0_B No receive interrupt pending 1_B Receive interrupt pending
TFO	30	rh	**Transmit FIFO Overflow Flag** Signals an overflow error. If enabled, an error interrupt is triggered. 0_B No overflow error pending 1_B Overflow error pending
TFL	31	rh	**Transmit FIFO Level Flag** A TXFIFO refill interrupt (if enabled) is generated when the filling level falls to INTLEVEL or below, each time when a data byte is taken out of the TXFIFO. This flag signals that the event from above occured. 0_B No transmit interrupt pending 1_B Transmit interrupt pending

Field	Bits	Type	Description
0	4, [12:7], 29	r	**Reserved** Read as 0; should be written with 0.

FLAGSSET

The FLAGSSET register contains the write only bits used to set the corresponding bits in the FLAGS register by software. Setting a flag bit triggers an interrupt, if the corresponding interrupt enable bit is set.

(>> **Table 19-4** register overview)

(>> **LIN Master Sequences**, **LIN Slave Sequences**)

(>> **FLAGS** register)

FLAGSSET
Flags Set Register **(38_H)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TFLS	**TFOS**	**0**	**RFLS**	**RFUS**	**RFOS**	**CES**	**LCS**	**LAS**	**LPS**	**BDS**	**RTS**	**HTS**	**FES**	**TCS**	**PES**
w	w	r	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRRQS	**THRQS**	**TWRQS**	**0**						**REDS**	**FEDS**	**0**	**RRS**	**RHS**	**TRS**	**THS**
w	w	w	r						w	w	r	w	w	w	w

Field	Bits	Type	Description
THS	0	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag
TRS	1	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag
RHS	2	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag

Field	Bits	Type	Description
RRS	3	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag
FEDS	5	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag
REDS	6	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag
TWRQS	13	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag
THRQS	14	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag
TRRQS	15	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag
PES	16	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag

Field	Bits	Type	Description
TCS	17	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag
FES	18	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag
HTS	19	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag
RTS	20	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag
BDS	21	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag
LPS	22	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag
LAS	23	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag

Field	Bits	Type	Description
LCS	24	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag
CES	25	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag
RFOS	26	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag
RFUS	27	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag
RFLS	28	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag
TFOS	30	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag
TFLS	31	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Set the corresponding flag
0	4, [12:7], 29	r	**Reserved** Read as 0; should be written with 0.

FLAGSCLEAR

The FLAGSCLEAR register contains the write only bits used to clear the corresponding bits in the FLAGS register.

(>> **Table 19-4** register overview)

(>> **FLAGS** register)

FLAGSCLEAR
Flags Clear Register **($3C_H$)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TFLC	**TFOC**	**0**	**RFLC**	**RFUC**	**RFOC**	**CEC**	**LCC**	**LAC**	**LPC**	**BDC**	**RTC**	**HTC**	**FEC**	**TCC**	**PEC**
w	w	r	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRRQC	**THRQC**	**TWRQC**	**0**						**REDC**	**FEDC**	**0**	**RRC**	**RHC**	**TRC**	**THC**
w	w	w	r						w	w	r	w	w	w	w

Field	Bits	Type	Description
THC	0	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clear the corresponding flag
TRC	1	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clear the corresponding flag
RHC	2	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clear the corresponding flag

Field	Bits	Type	Description
RRC	3	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clear the corresponding flag
FEDC	5	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clear the corresponding flag
REDC	6	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clear the corresponding flag
TWRQC	13	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clear the corresponding flag
THRQC	14	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clear the corresponding flag
TRRQC	15	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clear the corresponding flag
PEC	16	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clears the interrupt for the corresponding flag

Field	Bits	Type	Description
TCC	17	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clears the interrupt for the corresponding flag
FEC	18	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clears the interrupt for the corresponding flag
HTC	19	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clears the interrupt for the corresponding flag
RTC	20	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clears the interrupt for the corresponding flag
BDC	21	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clears the interrupt for the corresponding flag
LPC	22	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clears the interrupt for the corresponding flag
LAC	23	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clears the interrupt for the corresponding flag

Field	Bits	Type	Description
LCC	24	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clears the interrupt for the corresponding flag
CEC	25	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clears the interrupt for the corresponding flag
RFOC	26	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clears the interrupt for the corresponding flag
RFUC	27	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clears the interrupt for the corresponding flag
RFLC	28	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clears the interrupt for the corresponding flag
TFOC	30	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clears the interrupt for the corresponding flag
TFLC	31	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action 1_B Clears the interrupt for the corresponding flag
0	4, [12:7], 29	r	**Reserved** Read as 0; should be written with 0.

FLAGSENABLE

The FLAGSENABLE register contains the read write bits that enable the error interrupt in case the corresponding event has occurred.

(>> **Table 19-4** register overview)

(>> **LIN Master Sequences**, **LIN Slave Sequences**)

(>> **FLAGS** register)

FLAGSENABLE
Flags Enable Register **(40_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TFLE	**TFOE**	**0**	**RFLE**	**RFUE**	**RFOE**	**CEE**	**LCE**	**LAE**	**LPE**	**BDE**	**RTE**	**HTE**	**FEE**	**TCE**	**PEE**
rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0									**REDE**	**FEDE**	**0**	**RRE**	**RHE**	**TRE**	**THE**
r									rw	rw	r	rw	rw	rw	rw

Field	Bits	Type	Description
THE	0	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
TRE	1	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
RHE	2	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled

Field	Bits	Type	Description
RRE	3	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
FEDE	5	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
REDE	6	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
PEE	16	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
TCE	17	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
FEE	18	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
HTE	19	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled

Field	Bits	Type	Description
RTE	20	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
BDE	21	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
LPE	22	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
ABE	23	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
LCE	24	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
CEE	25	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
RFOE	26	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled

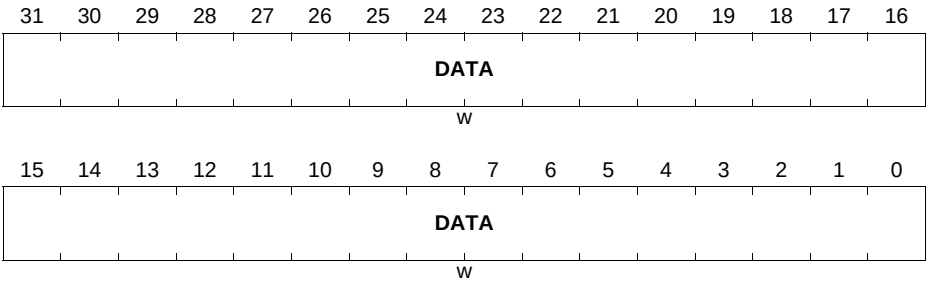
Field	Bits	Type	Description
RFUE	27	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
RFLE	28	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
TFOE	30	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
TFLE	31	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
0	4, [15:7], 29	r	**Reserved** Read as 0; should be written with 0.

TXDATA

Writing data to this register enters the data to the TXFIFO.

(>> **Table 19-4**)

TXDATA
Transmit Data Register **(44_H)** **Reset Value: 0000 0000_H**



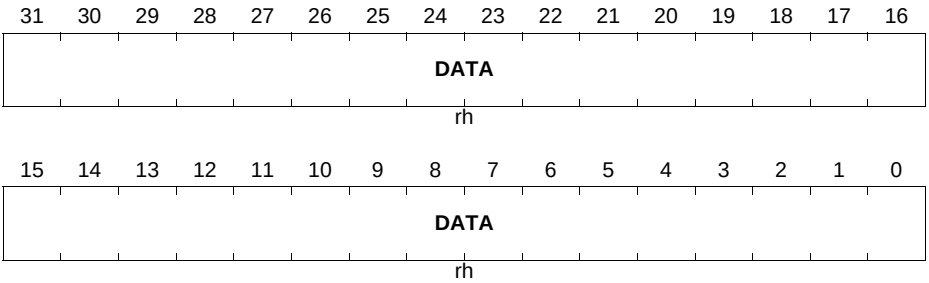
Field	Bits	Type	Description
DATA	[31:0]	w	**Data** Writing to this bit field writes the content to the TXFIFO, depending on the write width - 8, 16 or 32 bit. read from this register returns 0.

RXDATA

Reading data from this register takes data from the RXFIFO.

(>> **Table 19-4**)

RXDATA
Receive Data Register **(48_H)** **Reset Value: 0000 0000_H**



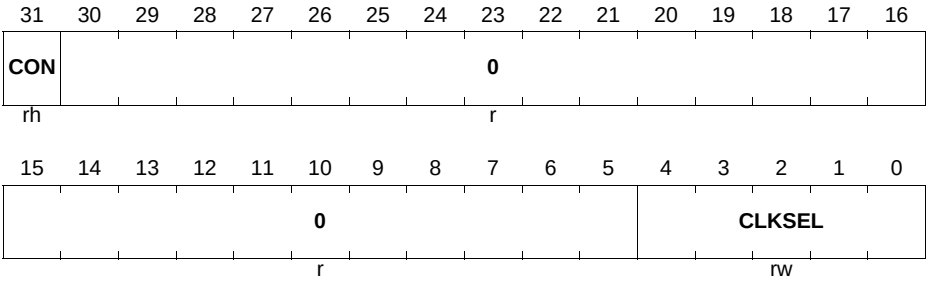
Field	Bits	Type	Description
DATA	[31:0]	rh	**Data** Reading from this bit field takes content from the RXFIFO, depending on the read width - 8, 16 or 32 bit. Write to this register has no effect.

CSR

This register is used to select the clock source for the baud rate generation, detection, timeouts and the SPI delays.

(>> **Table 19-4**)

CSR
Clock Selection Register **($4C_H$)** **Reset Value: 0000 0000$_H$**



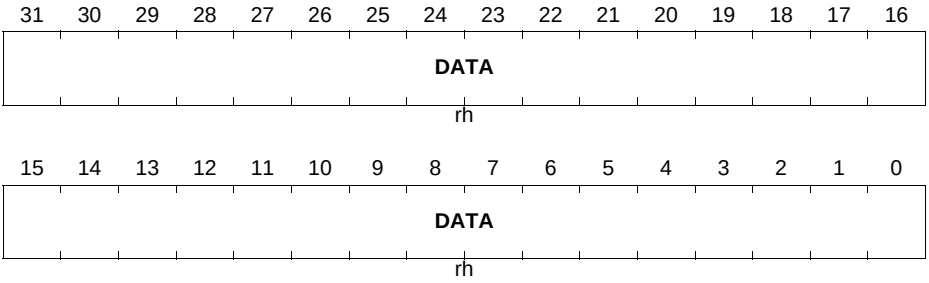
Field	Bits	Type	Description
CLKSEL	[4:0]	rw	**Baud Rate Logic Clock Select** 00000_B No clock supplied 00001_B f_{CLC} 00010_B XTAL Oscillator Clock f_{OSC0} 00100_B f_{ERAY} 01000_B $f_{ASCLINF}$ 10000_B $f_{ASCLINS}$ $...{}_B$ not allowed
CON	31	rh	**Clock On Flag** Shows if the clock in the bit time domain is switched on or off. Many configuration registers can be written only if this bit shows 0 (see **Table 19-4**). 0_B clock is off 1_B clock is on
0	[30:5]	r	**Reserved** Read as 0; should be written with 0.

RXDATAD

Reading data from this register takes data from the RXFIFO, but does not influence the read pointer. This virtual register provides non-destructive read to the RXFIFO.

(>> **Table 19-4**)

RXDATAD
Receive Data Debug Register (50_H) Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
DATA	[31:0]	rh	**Data** Reading from this bit field takes content from the RXFIFO, depending on the read width - 8, 16 or 32 bit, but dooes not influence the read pointer of the RXFIFO. Write to this register has no effect.

Clock Reconfiguration

The reconfiguration of the clock source has to be done by using two writes: first a write of zero to the CLKSEL bitfield, and then a second write defining the new clock source. Between the first and the second write a delay of minimum $T_W \geq 4 * (1/f_A) + 2 * (1/f_{CLC})$ must be inserted by software, where f_A is the frequency being switched off with the first write. Exception: in case that the f_{CLC} is selected as a baud rate and bit timing logic clock (CSR.CLKSEL = 1), no delay cycles between the writes are necessary. In both cases, simply using one write defining the new clock source is not allowed.

Additionaly, always activate the asynchronous clock for at least two f_A cycles:

- after entering the INIT state and
- after device reset

This is an example of a correct sequence:

- CSR.CLKSEL = No_Clock (equals the reset state)
- Wait T_W or poll for CSR.CON = 0

- FRAMECON.MODE = INIT
- CSR.CLKSEL = Clock_On
- Wait T_W or poll for CSR.CON = 1
- CSR.CLKSEL = No_Clock
- Wait T_W or poll for CSR.CON = 0
- FRAMECON.MODE = ASC (or SPI or LIN)
- CSR.CLKSEL = Clock_On
- Wait T_W or poll for CSR.CON = 1

See also **LIN Master Sequences**

See also **LIN Slave Sequences**

Abort Sequence

If a header has been transmitted in ASCLIN master mode, and the corresponding response does not come, the waiting for the response can be aborted by first entering the INIT mode and then entering the LIN mode. The sequence is identical with the example sequence above.

19.19 Implementation

This section describes the product specific configuration of the ASCLIN module and its interconnection with the rest of the system. The TC21x/TC22x/TC23x contains 2 interfaces, ASCLIN0 and ASCLIN1.

19.19.1 BPI_FPI Module Registers

19.19.1.1 System Registers

Figure 19-33 shows all registers associated with the BPI_FPI module, configured for one kernel.

BPI_FPI Registers Overview

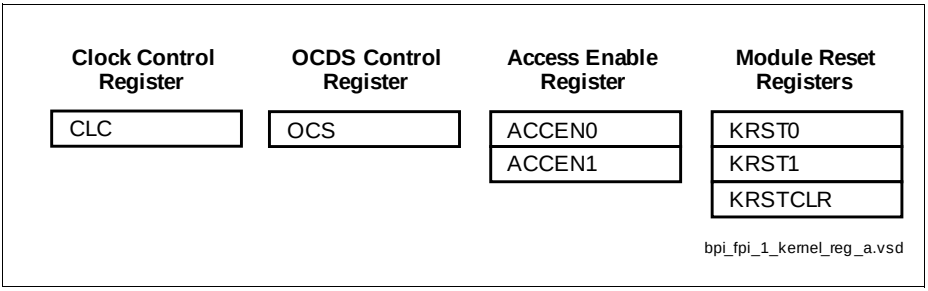


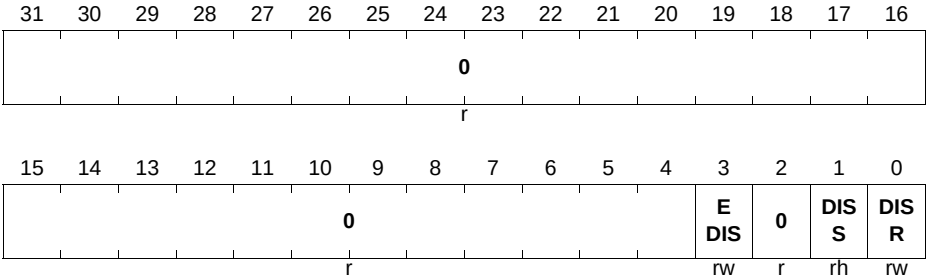
Figure 19-33 BPI_FPI Registers

Clock Control Register (CLC)

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI_FPI for the module. Where a module kernel is connected to the CLC clock control interface, CLC controls the f_A module clock signal, sleep mode and disable mode for the module.

(>> **Table 19-4** register overview)

CLC
Clock Control Register **(00_H)** **Reset Value: 0000 0003$_H$**



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module.
EDIS	3	rw	**Sleep Mode Enable Control** Used to control module's sleep mode, that is if the sensitivity of the module to the sleep signal is enabled to react to it or disabled to ignore it. 0_B Enabled 1_B Disabled
0	[31:4], 2	r	**Reserved** Read as 0; must be written with 0.

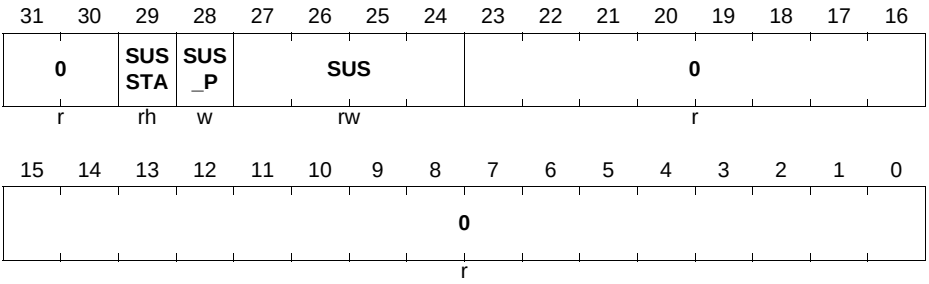
OCDS Control and Status Register (OCS)

The OCDS Control and Status (OCS) register is cleared by Debug Reset.

The register can only be written when the OCDS is enabled. If OCDS is being disabled, the OCS value will not change. When OCDS is disabled the OCS suspend control is ineffective. Write access is 32 bit wide only and requires Supervisor Mode

(>> **Table 19-4** register overview)

OCS
OCDS Control and Status **($E8_H$)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
SUS	[27:24]	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0_H Will not suspend 1_H Hard suspend. Clock is switched off immediately. 2_H Soft suspend **others**, reserved,
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	[23:0], [31:30]	r	**Reserved** Read as 0; must be written with 0.

Access Enable Register (ACCEN0)

The Access Enable Register 0 controls write access[1)] for transactions with the on chip bus master TAG ID 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000_B, EN1 -> TAG ID 000001_B, ... , EN31 -> TAG ID 011111_B.

(>> **Table 19-4** register overview)

ACCEN0
Access Enable Register 0 **(FC_H)** **Reset Value: FFFF FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN 31	**EN 30**	**EN 29**	**EN 28**	**EN 27**	**EN 26**	**EN 25**	**EN 24**	**EN 23**	**EN 22**	**EN 21**	**EN 20**	**EN 19**	**EN 18**	**EN 17**	**EN 16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN 15	**EN 14**	**EN 13**	**EN 12**	**EN 11**	**EN 10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n = 0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

1) The BPI_FPI Access Enable functionality controls only write transactions to the CLC, OCS, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers

Access Enable Register (ACCEN1)

The Access Enable Register 1 controls write access[1)] for transactions with the on chip bus master TAG ID 100000_B to 111111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the AURIX devices.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001_B, ... , EN31 -> TAG ID 111111_B.

(>> **Table 19-4** register overview)

ACCEN1
Access Enable Register 1 **($F8_H$)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
0	[31:0]	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 0 (KRST0)

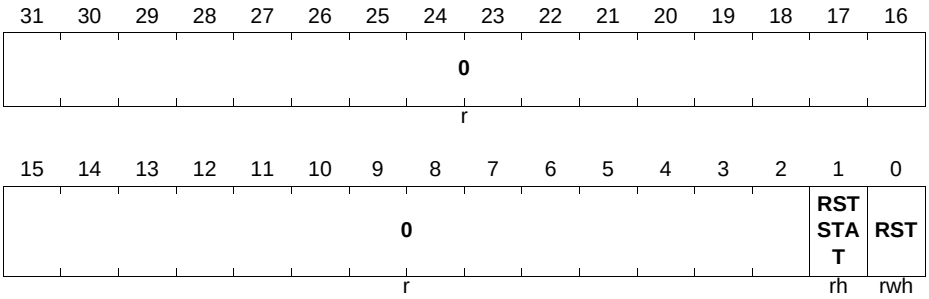
The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers.The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the KRSTCLE.CLR register bit.

Note: During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

(>> **Table 19-4** register overview)

KRST0
Kernel Reset Register 0 **($F4_H$)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed.

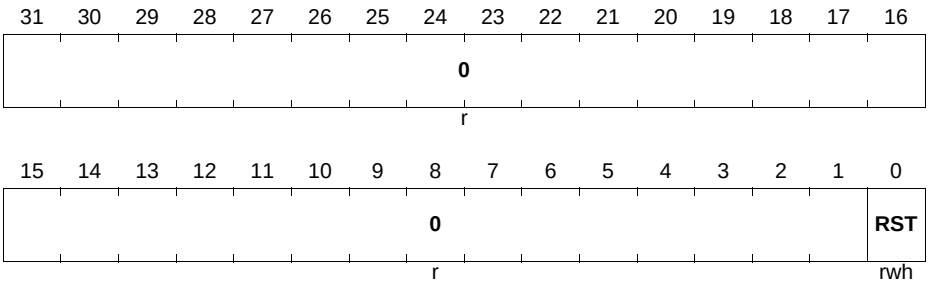
Field	Bits	Type	Description
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. 0_B No kernel reset was executed 1_B Kernel reset was executed This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register.
0	[31:2]	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1 (KRST1)

The Kernel Reset Register 1 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset registers. The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

(>> **Table 19-4** register overview)

KRST1
Kernel Reset Register 1 **($F0_H$)** **Reset Value: 0000 0000$_H$**



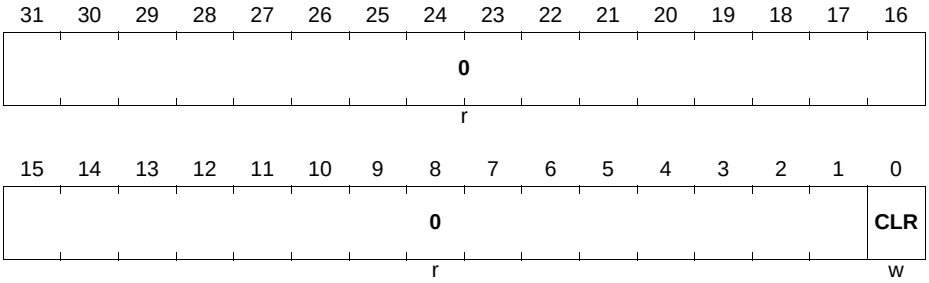
Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Status Clear Register (KRSTCLR)

The Kernel Reset Status Clear register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

(>> **Table 19-4** register overview)

KRSTCLR
Kernel Reset Status Clear Register (EC_H) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT Read always as 0.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

19.20 On-Chip Connections

This section describes the on-chip connections of the ASCLINx module instances.

Port/Pin Connections

The connections of the ASCLIN modules to the pins / ports is described in the chapters regarding the pinning and the ports.

Input signals not connected to ports/pins are connected to the following voltage levels:

- The unconnected ARX signals in the lower half range (RHA to RXD) are connected to (active) low level, “0”
- The unconnected ARX signals in the upper half range (RXE to RXH) are connected to (inactive) high level, “1”
- The unconnected ACTS signals are connected to low level “0”, which is after reset the inactive level (but disabled via **IOCR**.CTSEN bit field, meaning don’t care)

Table 19-5 shows an overview for TC23x how bits and bit fields must be programmed for the required I/O functionality of the ASCLIN I/O lines.

Table 19-5 ASCLIN I/O Control Selection and Setup

ASCLIN Module	Port Lines	Input Select Register	Input/Output Control Register Bits	I/O
ASC0	ARX0A / P14.1	ASC_IOCR.ALTI = 000_B	P14_IOCR0.PC1 = $0XXXX_B$	I
	ARX0B / P15.3	ASC_IOCR.ALTI = 001_B	P15_IOCR0.PC3 = $0XXXX_B$	I
	ARX0C / P0.1	ASC_IOCR.ALTI = 010_B	P00_IOCR0.PC1 = $0XXXX_B$	I
	ARX0D / P34.2	ASC_IOCR.ALTI = 011_B	P34_IOCR0.PC0 = $0XXXX_B$	I
	ACTS0A/ P0.12	ASC_IOCR.CTS = 00_B	P00_IOCR12.PC12 = $0XXXX_B$	I
	ARTS0 / P0.9	-	P00_IOCR8.PC9 = $1X011_B$	O
	ARTS0 / P14.7	-	P14_IOCR4.PC7 = $1X010_B$	O
	ATX0 / P0.0	-	P00_IOCR0.PC0 = $1X011_B$	O
	ATX0 / P0.1	-	P00_IOCR0.PC1 = $1X010_B$	O

Table 19-5 ASCLIN I/O Control Selection and Setup (cont'd)

ASCLIN Module	Port Lines	Input Select Register	Input/Output Control Register Bits	I/O
ASC0	ATX0 / P14.0	-	P14_IOCR0.PC0 = $1X010_B$	O
	ATX0 / P14.1	-	P14_IOCR0.PC1 = $1X010_B$	O
	ATX0 / P15.2	-	P15_IOCR0.PC2 = $1X010_B$	O
	ATX0 / P15.3	-	P15_IOCR0.PC3 = $1X010_B$	O
	ATX0 / P34.1	-	P34_IOCR0.PC0 = $1X010_B$	O
	ASCLK0 / P0.0	-	P00_IOCR0.PC0 = $1X010_B$	O
	ASCLK0 / P0.2	-	P00_IOCR0.PC2 = $1X010_B$	O
	ASCLK0 / P14.0	-	P14_IOCR0.PC0 = $1X110_B$	O
	ASCLK0 / P15.2	-	P15_IOCR0.PC2 = $1X110_B$	O
ASC1	ARX1A / P15.1	ASC_IOCR.ALTI = 000_B	P15_IOCR0.PC1 = $0XXXX_B$	I
	ARX1B / P15.5	ASC_IOCR.ALTI = 001_B	P15_IOCR4.PC5 = $0XXXX_B$	I
	ARX1C / P20.9	ASC_IOCR.ALTI = 010_B	P20_IOCR8.PC9 = $0XXXX_B$	I
	ARX1D / P14.8	ASC_IOCR.ALTI = 011_B	P14_IOCR8.PC8 = $0XXXX_B$	I
	ARX1E / P11.10	ASC_IOCR.ALTI = 100_B	P11_IOCR8.PC10 = $0XXXX_B$	I
	ARX1F / P33.6	ASC_IOCR.ALTI = 101_B	P33_IOCR4.PC6 = $0XXXX_B$	I
	ARX1G / P02.3	ASC_IOCR.ALTI = 110_B	P02_IOCR0.PC3 = $0XXXX_B$	I
	ACTS1A / P20.7	ASC_IOCR.CTS = 00_B	P20_IOCR4.PC7 = $0XXXX_B$	I

Table 19-5 ASCLIN I/O Control Selection and Setup (cont'd)

ASCLIN Module	Port Lines	Input Select Register	Input/Output Control Register Bits	I/O
ASC1	ACTS1B / P34.3	ASC_IOCR.CTS = 01_B	P34_IOCR0.PC3 = $0XXXX_B$	I
	ATX1 / P02.2	-	P14_IOCR0.PC0 = $1X010_B$	O
	ATX1 / P11.12	-	P11_IOCR12.PC12 = $1X010_B$	O
	ATX1 / P15.0	-	P15_IOCR0.PC0 = $1X010_B$	O
	ATX1 / P15.1	-	P15_IOCR0.PC1 = $1X010_B$	O
	ATX1 / P15.4	-	P15_IOCR4.PC4 = $1X010_B$	O
	ATX1 / P15.5	-	P15_IOCR4.PC5 = $1X010_B$	O
	ATX1 / P20.10	-	P20_IOCR8.PC10 = $1X010_B$	O
	ATX1 / P33.6	-	P33_IOCR4.PC6 = $1X100_B$	O
	ATX1 / P33.12	-	P33_IOCR12.PC12 = $1X010_B$	O
	ARTS1 / P20.6	-	P20_IOCR4.PC6 = $1X010_B$	O
	ARTS1 / P23.1	-	P23_IOCR0.PC1 = $1X010_B$	O
	ASCLK1 / P15.0	-	P15_IOCR0.PC0 = $1X110_B$	O
	ASCLK1 / P20.10	-	P20_IOCR8.PC10 = $1X110_B$	O
	ASCLK1 / P33.11	-	P33_IOCR8.PC11 = $1X010_B$	O
	ASCLK1 / P33.12	-	P33_IOCR12.PC12 = $1X100_B$	O
	ASLSO1 / P20.8	-	P20_IOCR8.PC8 = $1X010_B$	O

Table 19-5 ASCLIN I/O Control Selection and Setup (cont'd)

ASCLIN Module	Port Lines	Input Select Register	Input/Output Control Register Bits	I/O
ASC1	ASLSO1 / P33.10	-	P33_IOCR8.PC10 = $1X100_B$	O
	ASLSO1 / P14.3	-	P14_IOCR0.PC3 = $1X100_B$	O

ASCLIN Connection to Itself

the following connection is defined for each ASCLIN instance:

ARTS -> ACTSD

This connection can be useful in the SPI mode, where frequently the transmission and the reception of data are performed in parallel. If this connection is selected by using **IOCR**.CTS, the TXFIFO will deliver data for transmission only if there is a free space in the RXFIFO. If not, the TXFIFO will wait for emtying of the RXFIFO by software and then automaticaly continue the transmission.

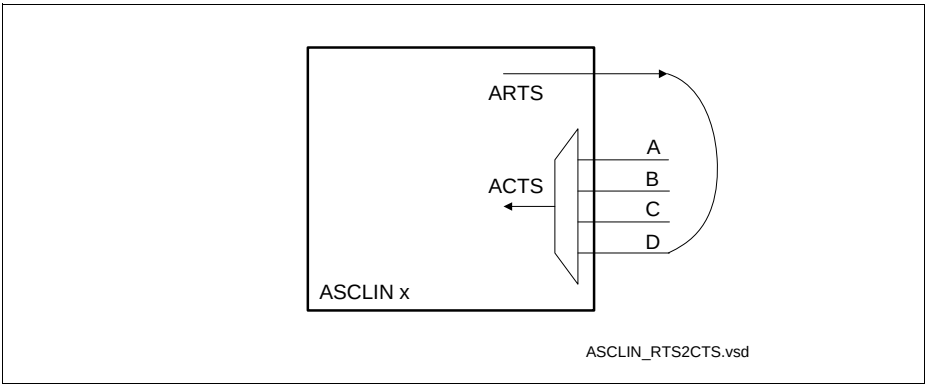


Figure 19-34 ASCLIN ARTS to ACTS Connection

19.21 ASC at CAN Support

The ASC Tx and Rx signals are overlayed with CAN Rx and Tx signal. See the Ports chapter and the Pinning chapter for the particular pin assignements.

A virgin device with a completely erased flash memory can be booted via ASC at CAN pins. See the Boot chapter for details on the implementation.

20 Queued Synchronous Peripheral Interface (QSPI)

The main purpose of the QSPI module is to provide synchronous serial communication with external devices using clock, data-in, data-out and slave select signals. The focus of the module is set to fast and flexible communication: either point-to-point or master-to-many slaves communication.

Parallel requests from on chip bus masters to a module will be executed sequentially via the on chip bus system. Read-modify-write feature provides an atomic read/write sequence where no other master can access the module in between. Module hardware semaphores are not supported.

>> **Registers Overview**

>> **BPI_FPI Module Registers**

20.1 Feature List

This section describes the features of the QSPI module.

- Master and Slave Mode operation
  - Full-duplex operation
  - Half-duplex operation
  - Automatic slave select control
  - Four-wire and three-wire type of connection
- Flexible data format
  - Programmable number of data bits: 2 to 32 data bits (plus parity: 3 to 33 bits)
  - 4 to 32 data bits possible for 50 Mbit/s
  - Programmable shift direction: LSB or MSB shift first
  - Programmable clock polarity: Idle low or idle high state for the shift clock
  - Programmable clock phase: data shift with leading or trailing edge of the shift clock
- Baud rate generation
  - Flexible baud rate and delays (leading, trailing, idle) generation
- Interrupt generation
  - On a transmitter FIFO event
  - On a receiver FIFO event
  - On an error condition (receive, baud rate, transmit error, parity error)
  - On a phase transition (start of frame, end of frame ...)
- QSPI supports control and data handling by the DMA controller
- Flexible QSPI pin configuration
- Hardware supported parity mode
  - Odd / even / no parity
- Seven slave select inputs $\overline{\text{SLSIB..H}}$ in Slave Mode
- Sixteen programmable slave select outputs SLSO[15:0] in Master Mode
  - Automatic SLSO generation with programmable timing
  - Programmable active level and enable control

  - External demultiplexing of the slave select outputs support
- Several module reset options
  - State machine reset per software (only the state machine)
  - Module reset per software (both FIFOs, all registers and the state machine)
  - Automatic stop option of the state machine in slave mode after baud rate error
- Loop-Back mode
- Interoperability with SSC and USIC modules of Infineon microcontroller families, and with popular (Q)SPI interfaces of multiple suppliers
- Communication stop on RxFIFO full
  - Shift register full and RxFIFO full can pause the communication
  - Interrupt generation
- High Speed Input Capture (HSIC)
  - Provides input capture functionality for ADAS applications
  - 15-bit counter with resolution of f_{BAUD2}

20.2 Overview

This section gives a top-level overview of the QSPI.

20.2.1 External Signals

The communication between two devices using QSPI generally uses four signals:

- serial clock SCLK
- data in master to slave direction MTSR (Master Transmit Slave Receive)
- data in slave to master direction MRST (Master Receive Slave Transmit)
- slave select signal SLS

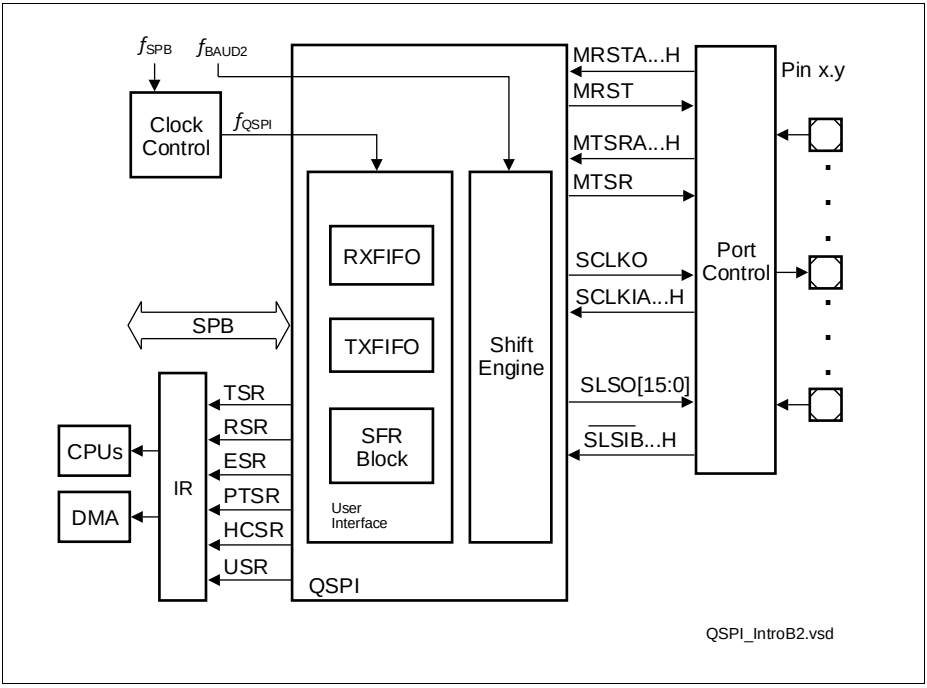


Figure 20-1 External Signals of the QSPI Module

20.2.2 Operating Modes

The QSPI operates in one of two modes regarding the generation of the serial clock and slave select signals: master or slave mode. The master generates and drives the clock and select signals, the slave receives these signals.

The QSPI operates in one of three modes regarding the direction of the communication and depending whether the transmission and the reception appear simultaneously or not: duplex, half-duplex and simplex mode.

This gives six possible combinations for the operating mode of the QSPI: master duplex, half-duplex, and simplex; slave duplex, half-duplex, and simplex.

Note: The half-duplex mode can be implemented either by short-cut connection between two different pins on the pcb, one data output and one data input pin, or by using single pins mapped to both data input and data output signals. See the pinning and port chapters for the pinning definition of a particular product. The module itself does not differentiate between the full-duplex, half-duplex or simplex connection.

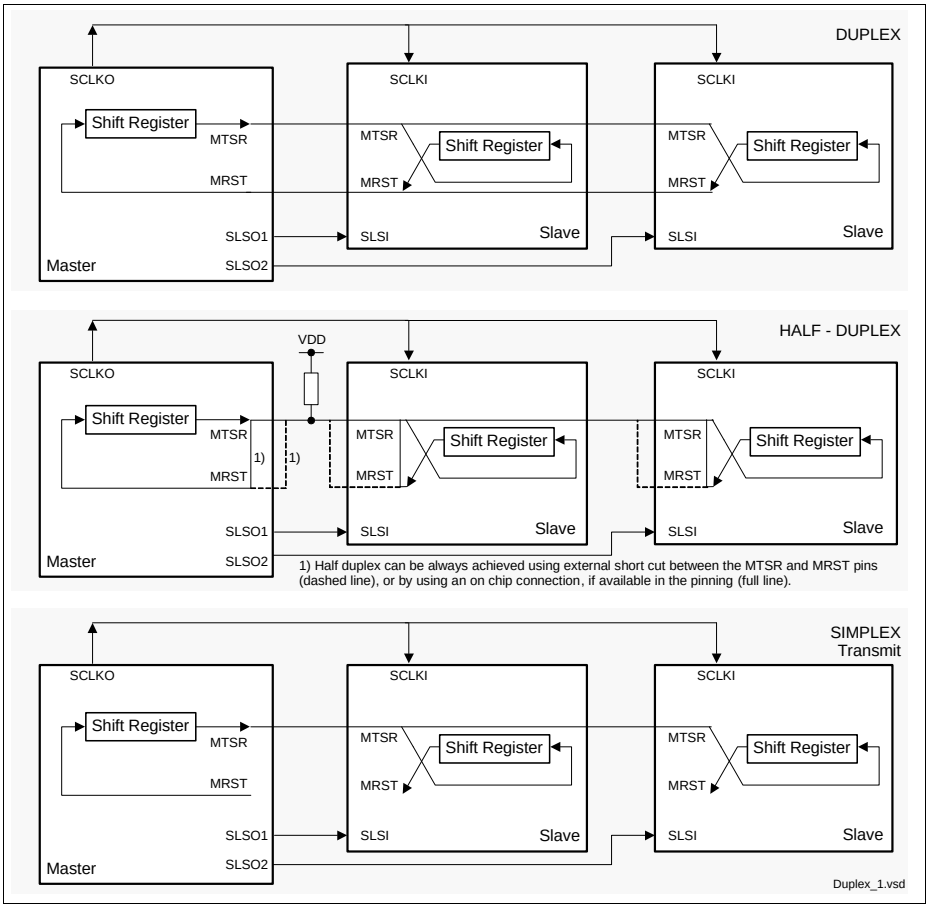


Figure 20-2 Operating Modes and Types of Connections

20.2.3 Queue Support Overview

The term "Queue Support" is used in this context to describe the functionality implemented for comfortable switching of the timing configuration of the QSPI frames, depending on the slave select signal which is to be activated. The main feature of the module is the possibility to take both the configuration and data to the TXFIFO, and to track down which TXFIFO entry is configuration, and which data. The QSPI module expects 32 basic configuration bits to be moved with one move (for example DMA move) from some on-chip general purpose RAM to the TXFIFO. These 32 configuration bits from the TXFIFO and the configuration bits from the 8 configuration extension registers **ECON**z contained in the QSPI module, define together the full configuration of the module (see **Figure 20-3**). One extension register is used for two slave selects: ECON0 for SLSO0 and SLSO8, ECON1 for SLSO1 and SLSO9 ...

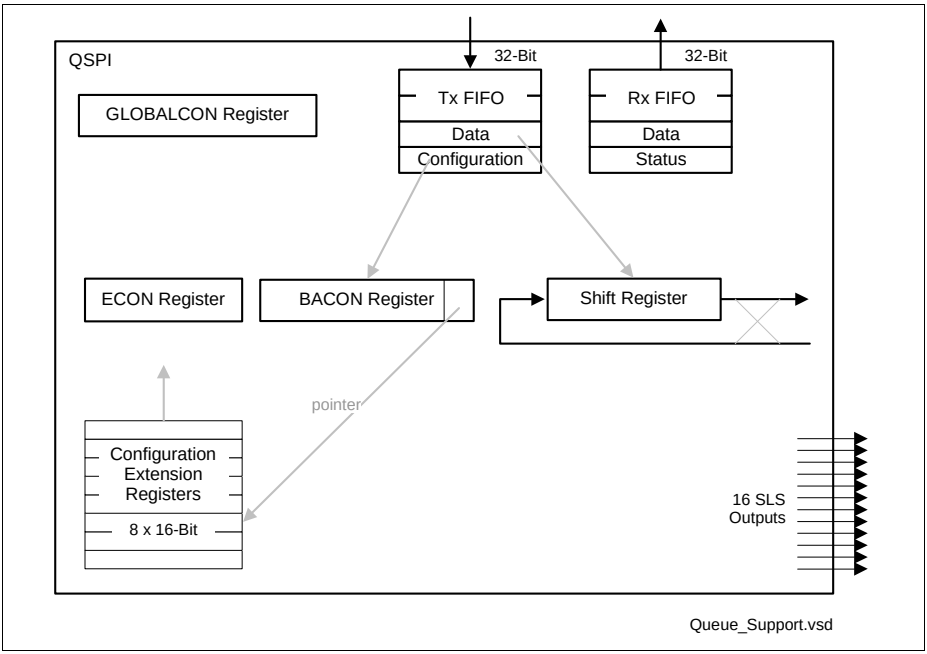


Figure 20-3 Queue Support Overview

20.2.4 Architecture Overview

The **Figure 20-4** shows the main blocks of the QSPI module. The TxFIFO and RxFIFO provide the user interface. The Shift Register and the “Miscellaneous Logic” block build the state machine of the module. The “Configuration Extensions” block provides comprehensive capabilities for configuring the QSPI frames.

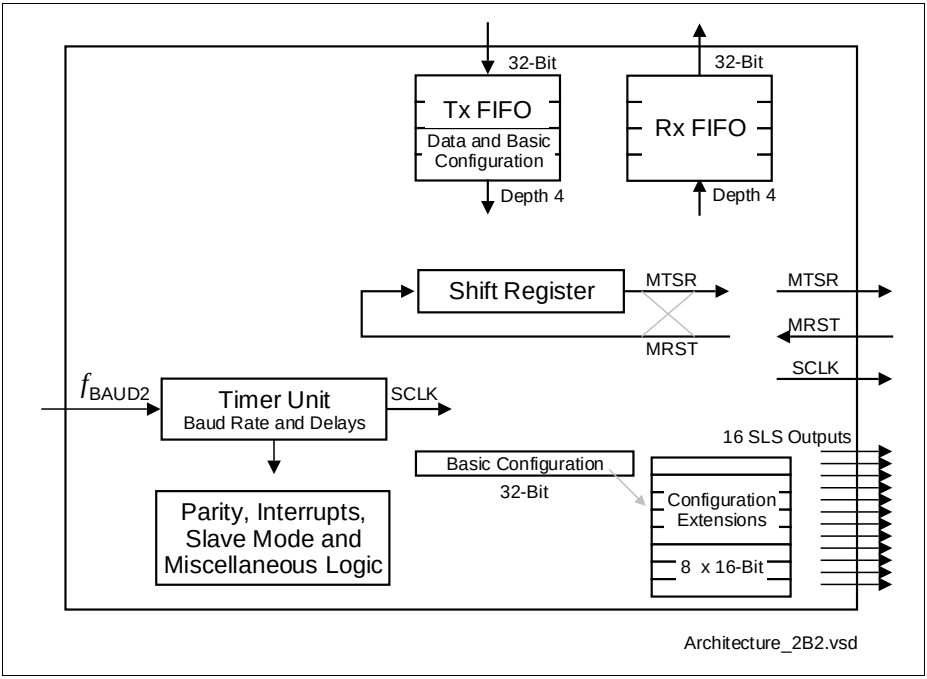


Figure 20-4 QSPI - Architecture Overview

20.2.5 Three Wire Connection

The term "Three Wire Connectiont" is used in this context to describe a connection without slave select signal. This connection relies solely on counting bits for determining the end of the current frame and resetting the shift register state machine, instead of using the slave select signal for this purpose. This way of communication is less robust than communication with slave select, but it saves a pin on both master and slave device.

The name “three wire connection” is true only in case of full-duplex connection, where exactly three signals (clock, data-in, data-out) are needed. In case of half-duplex and simplex connections only two signals (clock, data-in/data-out common data line) are necessary.

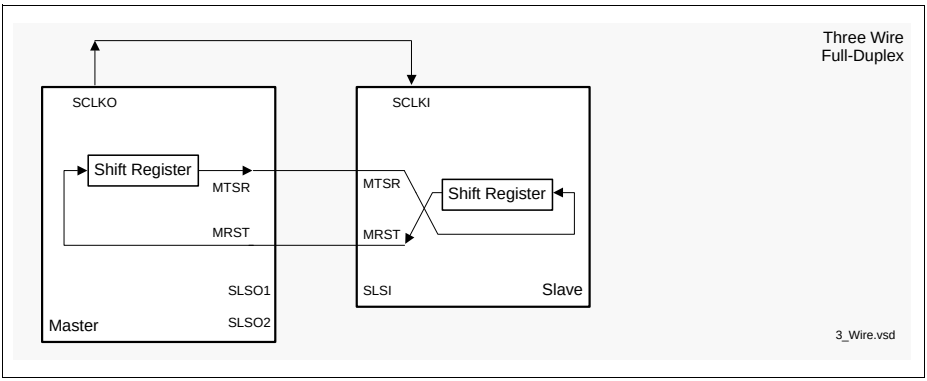


Figure 20-5 Three Wire Connection

20.3 Abstract Overview

An abstract overview of the QSPI module is shown in **Figure 20-6**. This document describes the QSPI module according to this view: the State Machine with its configuration and status capabilities, the User Interface, and the Interrupts (in this order).

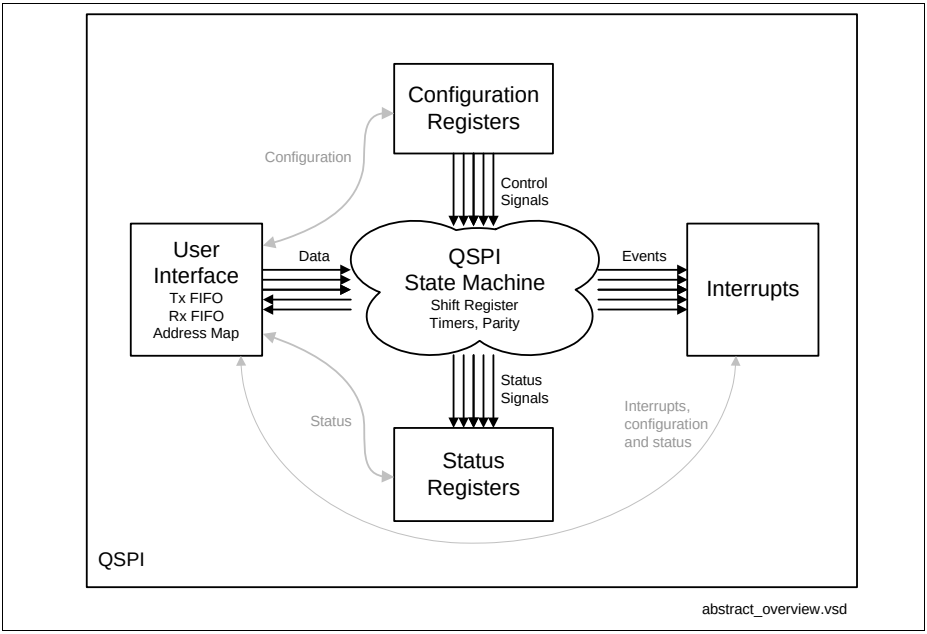


Figure 20-6 QSPI - Abstract Overview

20.4 Frequency Domains

The state machine of the QSPI is placed in a separate frequency domain operating with a frequency f_{BAUD2} which can be:

- integer multiple of the SPB bus frequency
- equal to the SPB frequency
- lower than the SPB frequency, derived by integer division

The baud rate is determined by the f_{BAUD2} frequency divided by an integer equal or greater than four.

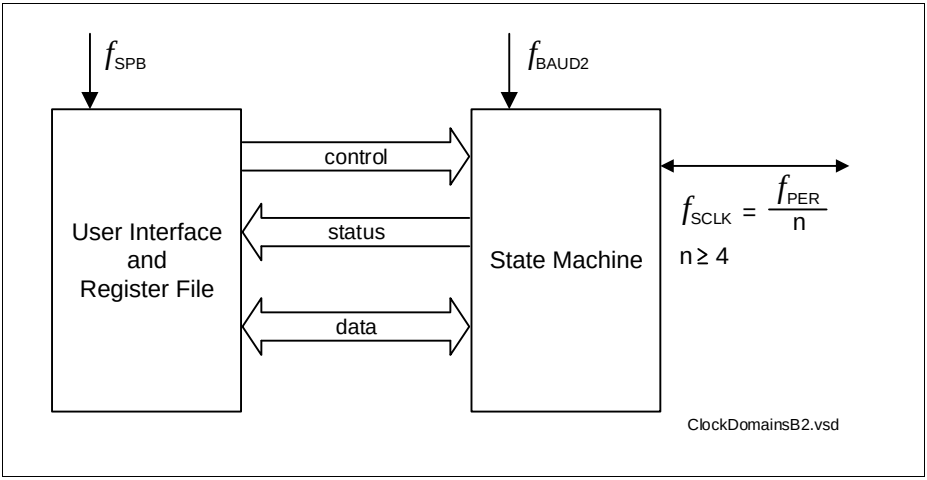


Figure 20-7 QSPI - Frequency Domains

20.5 Master Mode State Machine

In master mode, the QSPI module generates the timings, the serial clock, and the slave select signals.

20.5.1 Phases of one Communication Cycle

This section describes the possibilities available for configuring the length of the phases of the QSPI communication: timing delays, data length, duty cycle and data sampling.

A QSPI frame starts with activating a slave select signal SLSO (at transition from idle to leading delay phase), and ends with its deactivation (at transition from trailing delay to wait or idle phase). It is a sequence of five phases: idle delay, leading delay, data phase, trailing delay and an optional wait phase. The idle phase is subdivided in two phases of equal length: idle A and idle B.

Figure 20-8 shows a full and a compressed view of a QSPI frame and its phases. The full view shows all four signals needed for a QSPI connection. The compressed view represents the phases in a single row, in a way suitable to discuss their properties.

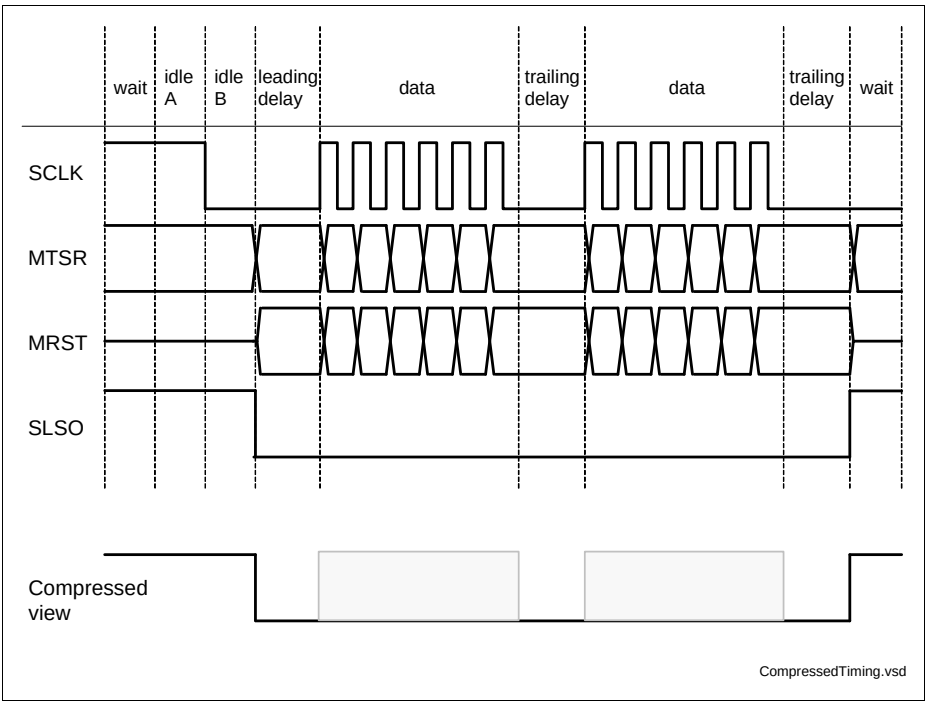


Figure 20-8 Phases of a QSPI Frame (Example for Data Length of 5 Bits)

The flexible timing control of the QSPI allows to program each phase in a very wide time range, with sufficient precision. Some examples of QSPI frames, in compressed view, with different lengths of the phases, are shown in **Figure 20-9**.

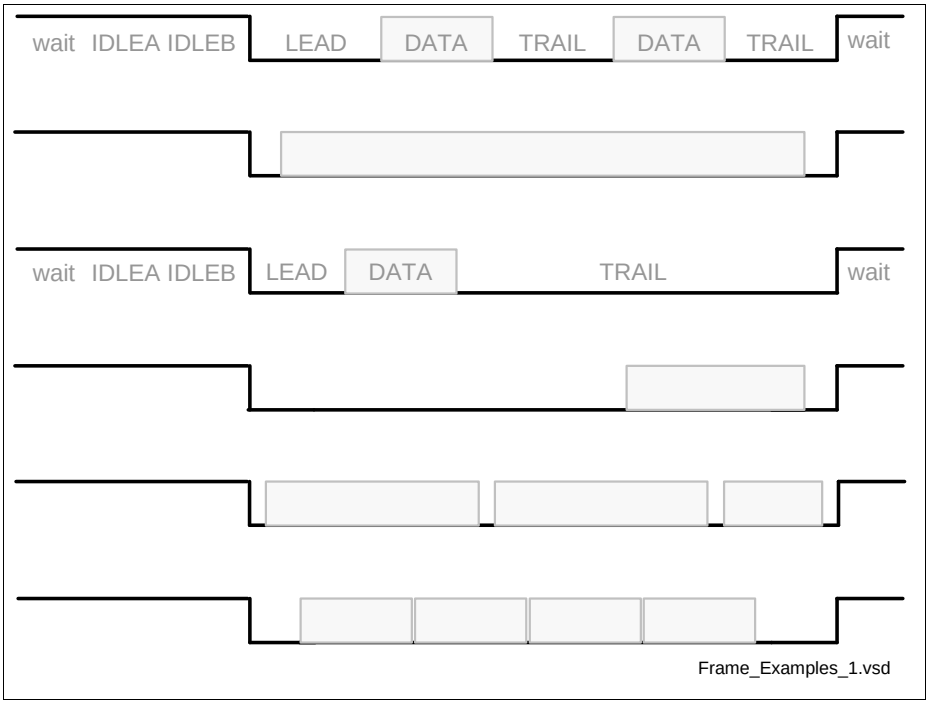


Figure 20-9 Examples of QSPI Frames in Compressed View

The length of the phases is programmed using the corresponding bit fields in the register **BACON**. There are four main bit fields defining the length of the phases in t_Q units and several modifier bit fields defining their range and granularity:

- IDLE
- LEAD
- DATA
- TRAIL

The WAIT phase is simply a loop waiting for a write to the Tx FIFO, without predefined duration, and consequently without a defining bit field.

The IDLE bit field defines two sub-phases with the same length IDLEA and IDLEB. At the transition between them the polarity of the serial clock signal for the next slave is set / changed.

In addition, the flexible timing control allows to program the duty cycle and the sampling point properties of the serial clock. This allows to improve to a certain extent the achievable baud rate taking into account the clock asymmetries and the loop-delay.

The **ECON** bit fields A, B and C define the length of the corresponding bit segments.

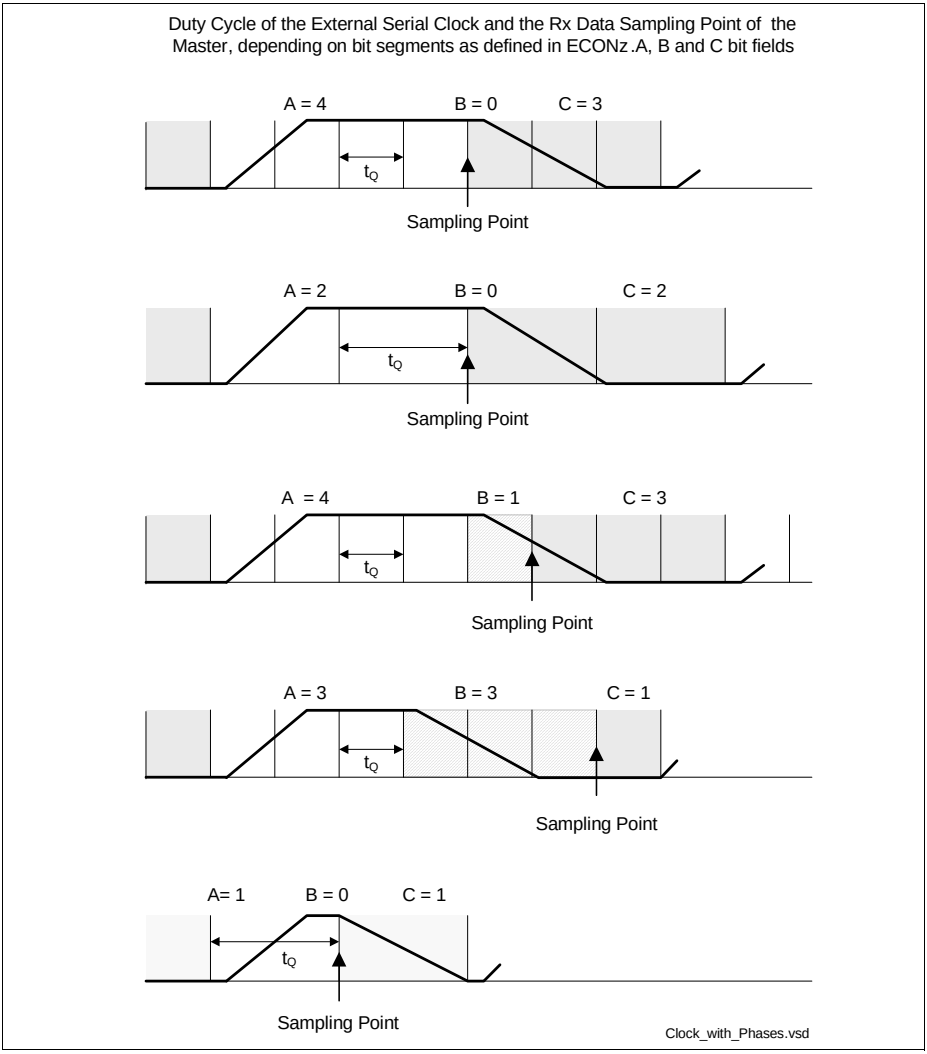


Figure 20-10 Control of the Duty Cycle and the Sampling Point of the Serial Clock

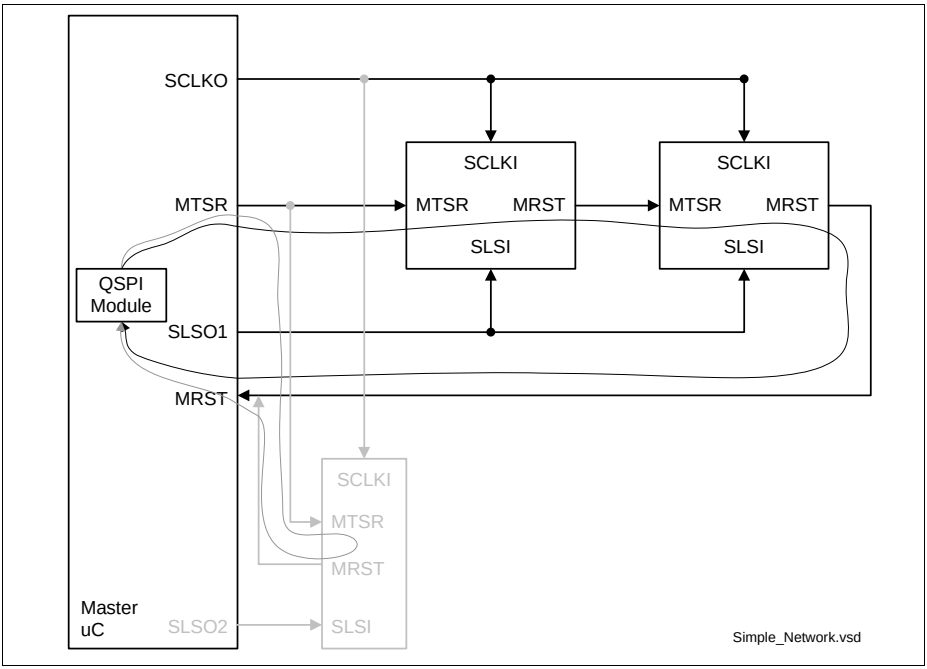


Figure 20-11 Closed-loop delays in a simple network consisting of one daisy-chain and one point to point connection in parallel

The purpose of the duty cycle control is:

- to compensate the influence of the path and pad asymmetry on the duty cycle
- to compensate the considerable asymmetry of the edges in case of half-duplex connection implemented as open-drain driver with pull-up resistor
- to adjust the duty cycle optimally to the timing properties of the master and slave, and to the properties of the connection (distance and capacitive load)
- to help achieve optimal baud rate, especially at higher baud rates which need an odd division factor (for example 5 = 3 high time + 2 low time or 5 = 2 high time + 3 low time)

The purpose of the sampling point control is to allow reliable reception of the data transmitted by the slave, at as high as possible baud rates.

In master receive direction the slave transmitted data reaches the master considerably late relative to the shift edge of the serial clock. This is caused by the loop delay consisting of:

- the traveling time of the shift clock edge from the master to the slave
- the reaction time of the slave
- the traveling time of the data from the slave to the master

Last but not least, the flexible timing control allows to control the phase and the polarity of the shift clock.

The clock polarity is configured by bit field **ECON**z.CPOL. This bit field sets the clock idle polarity to low or high.

The clock phase is configured by bit field **ECON**z.CPH. This bit field sets the initial clock delay to 0 or Bit Segment A, as defined by the bit field **ECON**z.A.

The CPOL and CPH settings control the master mode. The slave mode timing behavior is fixed to CPOL=0 and CPH=1, and these values must be programmed in the **ECON** register pointed by **BACON**.CS.

The **Figure 20-12** shows the QSPI timings for clock phase CPH = 0.

ECON.CPH = 0 is used when communicating with a slave that delivers a valid value of the first bit immediately after being selected with an SLS signal. In such a case the first clock edge in a frame is used to latch the first bit. The second edge delivers the second bit value, and so on. The last edge in a frame delivers a bit with don't care value.

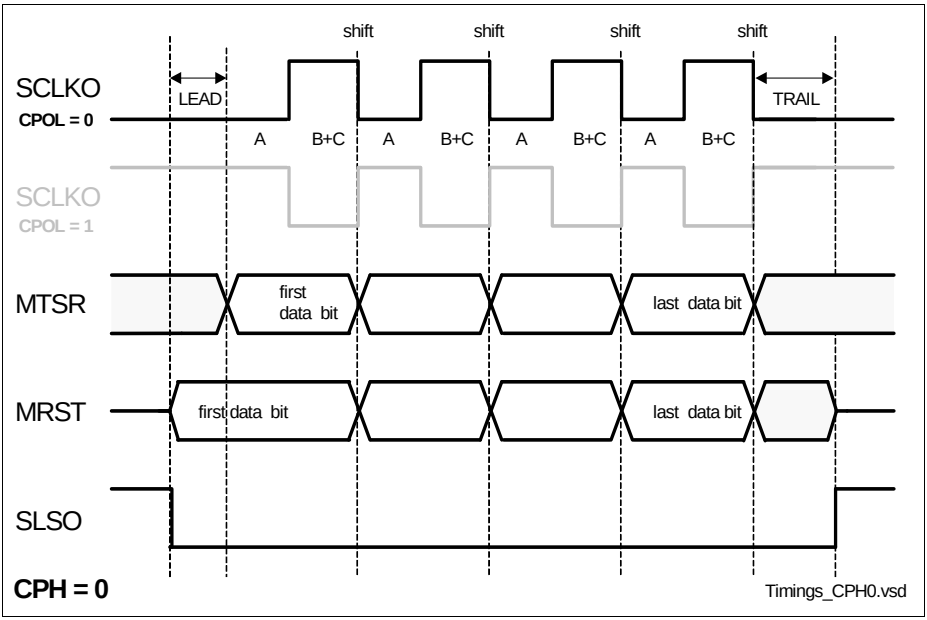


Figure 20-12 Timing of a transfer with CPH = 0 (Example for Data Length of 4 Bits)

The trailing delay starts after the last shift clock period of a data block and is followed either by the deactivating edge of SLSO, or a new data block in continuous mode.

Attention: If CPH = 0, then the total delay between the SLSO activating edge and the first edge of the serial clock SCLKO is LEAD + ECONz.A

Figure 20-13 shows the QSPI timings for clock phase CPH = 1.

ECON.CPH = 1 is used when communicating with a slave that delivers a bit with a random or don't care value immediately after being selected with the SLSO signal. In such a case the first clock edge in a frame is used by the slave to shift out the first valid data bit. The second edge latches this bit value, and so on. The last edge in a frame is used to latch the last valid data bit, which normally remains driven until the end of the frame.

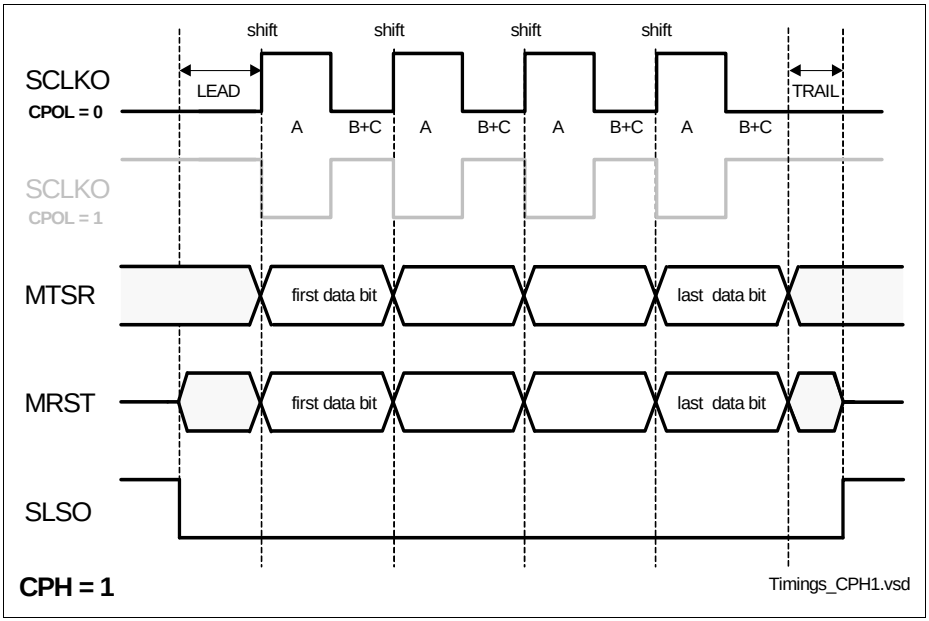


Figure 20-13 Timing of a transfer with CPH = 1 (Example for Data Length of 4 Bits)

The leading delay lasts between the active edge of the SLSO and the first shift clock edge.

Attention: If CPH = 1, then the total delay between the last edge of the SCLKO and the SLSO deactivating edge is TRAIL + ECONz.B + ECONz.C

20.5.2 Configuration Extensions

This section describes the possibilities available for configuring the phases of the QSPI communication: timing delays, data lengths, their ranges and granularity.

The QSPI module controls 16 communication channels, which are individually programmable. Each channel is associated to one slave select output and to the corresponding timing and other properties (baud rate, delays, data width, parity...).

The 16 slave select channels of the QSPI module are subdivided in two groups of 8 slave select channels:

- Channels 0 to 7
- Channels 8 to 15, channel 8 having the same **ECON** settings as channel 0, channel 9 as channel 1, and so on.

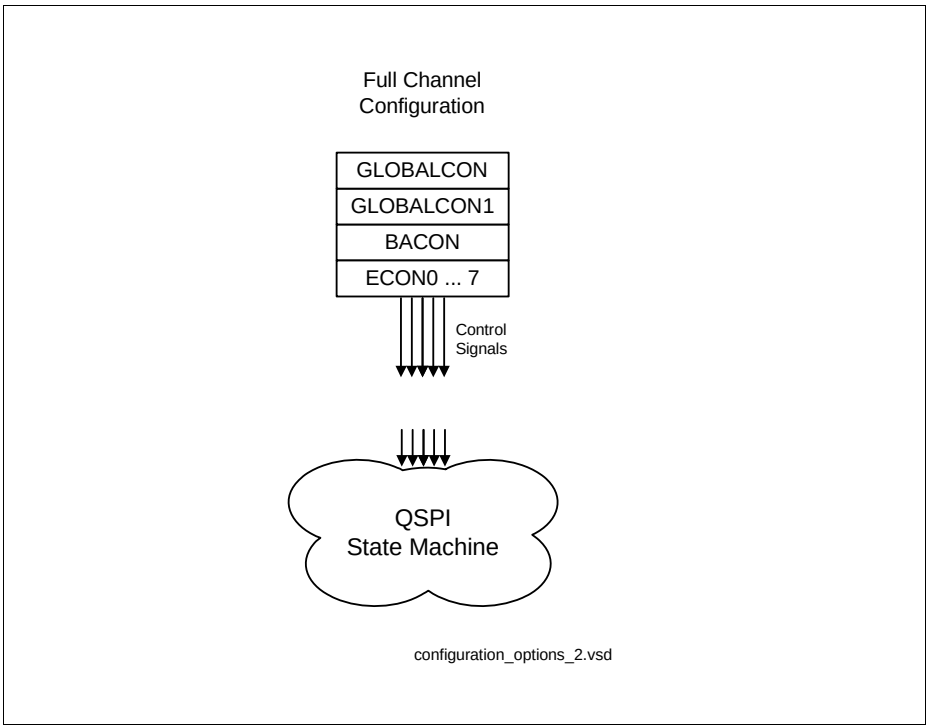


Figure 20-14 Flexible Configuration Concept

20.5.3 Details of the Baud Rate and Phase Duration Control

The QSPI phases correspond to states of a simple state machine defining a sequence of simple states and loops. Due to the strict sequential nature of the communication cycle one timer is enough to define the length of each phase. The length of each phase is defined in time quanta, and the length of a time quantum is defined by a time quantum timer. Each QSPI slave select channel has its own time quantum length, based on the module time quantum length:

T_{BAUD2} clock period -> one module time quantum -> one channel time quantum

Each QSPI module generates a module time quantum which is further downscaled to a channel quantum. This affects both the delays and the baud rates. The assumption is that the pad strength setting, the capacitive load and the corresponding QSPI network layout properties do not allow extreme differences in the baud rates for the different slaves connected to one module. They all have similar speed range: high speed, or medium speed, or low speed. Mixing very high speed and very low speed slaves does not make much sense. Many slaves produce high accumulated capacitive load on the clock and data outputs, which influences the possible baud rates for all slaves. Variations in the baud rates of the slaves of one module in the range of 6:1 is possible by varying the bit segment phases in each channel. Additional two-bit counter per slave allows for additional scaling of 1, 2, 4, or 8, making a total bit-time variation of 48:1 between the channels in one module possible.

Each QSPI slave select channel has its own set of phase lengths, depending on $T_{BAUD2} = 1 / f_{BAUD2}$

- $T_{SCLKz} = T_{BAUD2}$ * **GLOBALCON**.TQ * **ECON**z.Q * (A + B + C)
- $T_{LEAD} = T_{BAUD2}$ * **BACON**.LPRE * **BACON**.LEAD
- $T_{TRAIL} = T_{BAUD2}$ * **BACON**.TPRE * **BACON**.TRAIL
- $T_{IDLEA,B} = T_{BAUD2}$ * **BACON**.IPRE * **BACON**.IDLE
- $T_{STROBE} = T_{BAUD2}$ * **GLOBALCON**.TQ * **ECON**z.Q * **GLOBALCON**.STROBE

*Note: The equations above contain the representative decimal values of the bit-fields in corresponding units, not their actual binary bit-value. For the formulas using the actual binary values of the bit-fields, see **Figure 20-16** and **Figure 20-17**.*

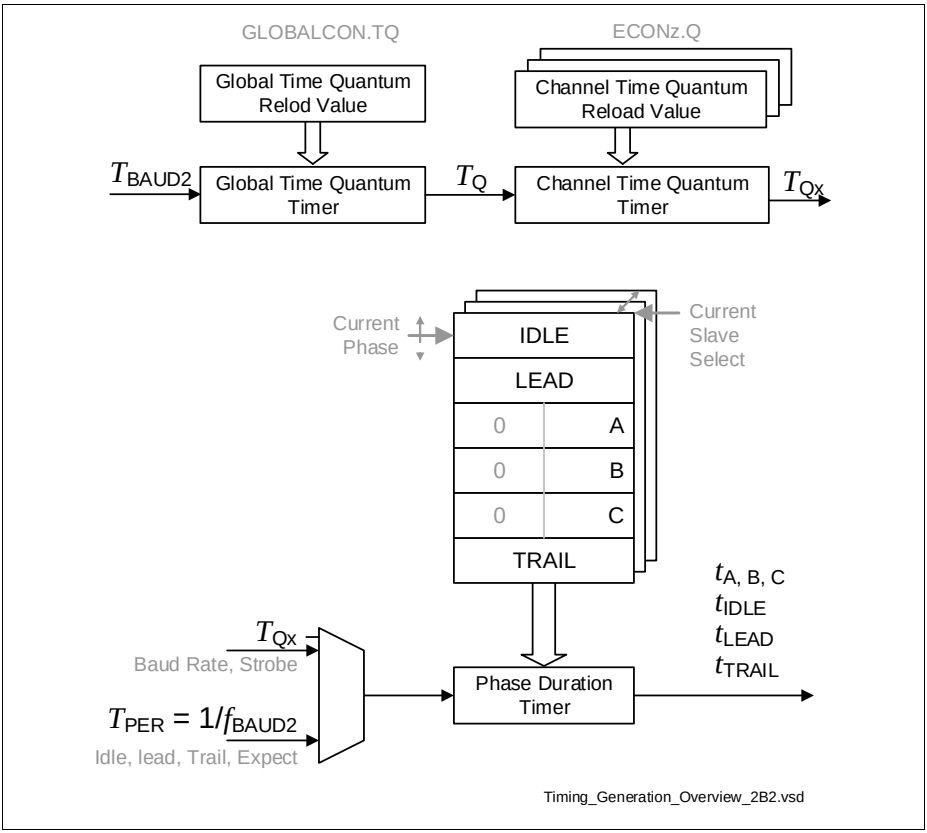


Figure 20-15 Phase Duration Control, Overview

20.5.4 Calculation of the Baud Rates and the Delays

This section gives another alternative view to the calculation of the baud rate and the duration of the delays.

The baud rate generation chain for a channel starts with a common TQ divider. The resulting global time quantum is used by the dedicated Q and A,B,C dividers to generate the baud rates per channel (slave select). See **Figure 20-16**.

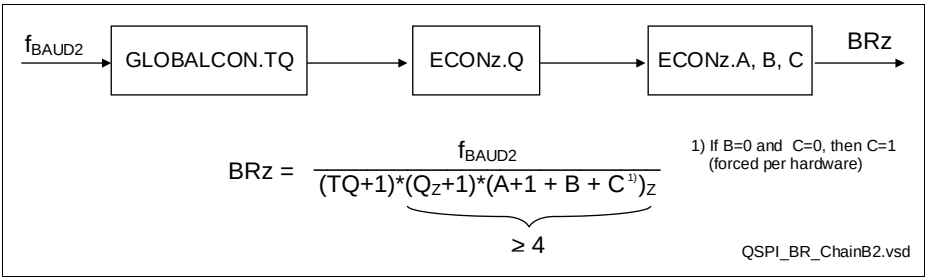


Figure 20-16 Baud Rate Calculation

Attention: The following condition must be satisfied: (Q+1)*(A+1 +B +C) >= 4. Or alternatively formulated: bit length must be at least 4*T_Q.

Each delay of an active channel (slave select) has a separate and independent divider chain starting with a power-of-four prescaler and an n-divider. See **Figure 20-17**.

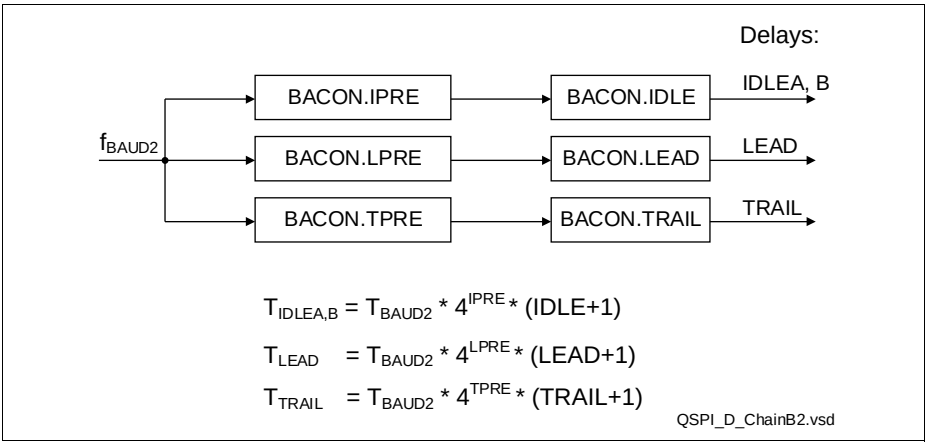


Figure 20-17 Calculation of the delays

20.5.5 State Diagram of Standard Communication Cycle

Figure 20-18 shows the top view of the standard communication cycle of the QSPI state machine in a hierarchical way. The “Data” state consists of a loop of several “Bit” states, where the number of repetitions is defined with the data width bit field. Each “Bit” state consists of a simple sequence of “A”, “B”, and “C” states.

A complete (or standard) communication cycle containing all phases, starting with activating the slave select and ending with its deactivation, is executed only if the bit **BACON**.LAST = 1. Otherwise, the slave select remains active and the module waits for more FIFO data entries.

The Phase Transition Interrupt PT (events PTI1 and PTI2) signal one out of all phase transitions which are shown in **Figure 20-18**, selected by the bit fields **GLOBALCON**.PT1 and PT2.

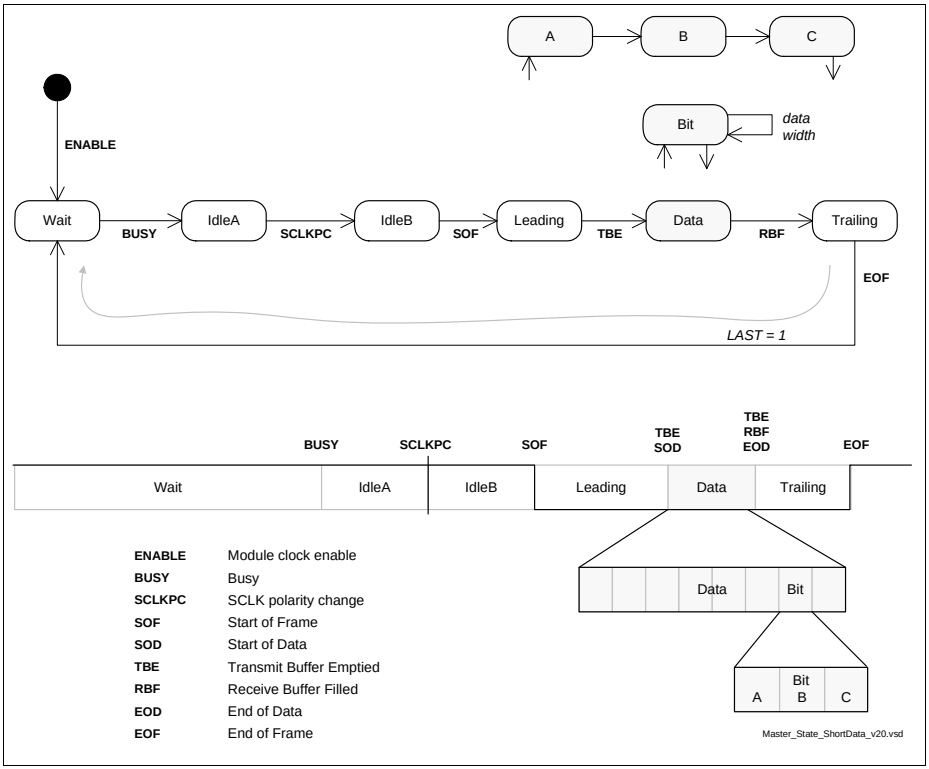


Figure 20-18 Standard Communication Cycle of the State Machine (Short Data)

(>>**Interrupts**)

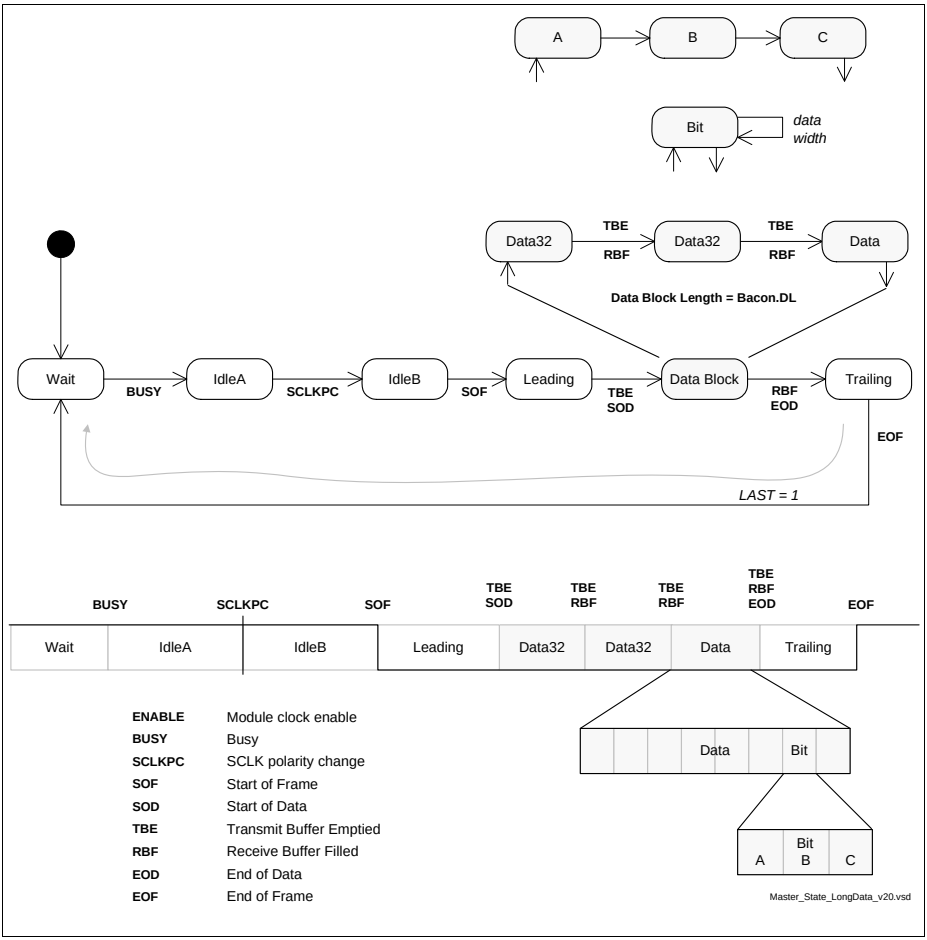


Figure 20-19 Standard Communication Cycle of the State Machine (Long Data)

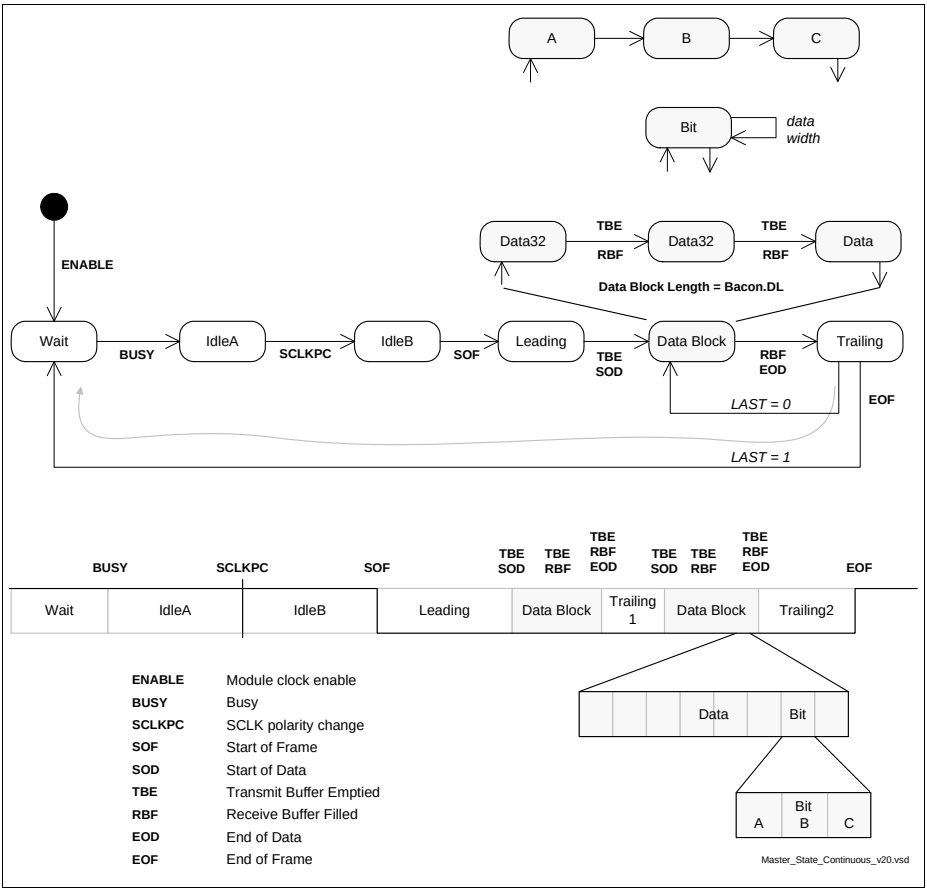


Figure 20-20 Standard Communication Cycle of the State Machine (Continuous)

20.5.6 Expect Phase

Figure 20-21 shows the EXPECT state in the QSPI state machine diagram. This state can be entered in the long data mode and in the continuous mode, where one frame consists of more than one TXFIFO entries. Its purpose is to provide monitoring of the latencies on the FPI bus. In this case, after one FIFO entry has been shifted out and the module expects more bits to shift, according to the **BACON**.DL, a time-out counting starts. If the expected FIFO entry does not come in time, a time-out occurs.

The duration of the EXPECT state is zero if at the end of the current DATA state the new data is already available in the FIFO.

The duration of the EXPECT state can be anywhere between zero and the programmed upper limit, if the next data is written into the TXFIFO any time within the allowed time window. In this case the current frame continues with the SLSO signal remaining active.

The duration of the EXPECT state is exactly maximum if the data did not come in time. Then the current frame is stopped, but the SLSO is not deactivated. A TO (Time - Out) interrupt is raised. The state machine continues the EXPECT state in a loop, waiting for the software intervention, generating an interrupt signal after each expect time interval.

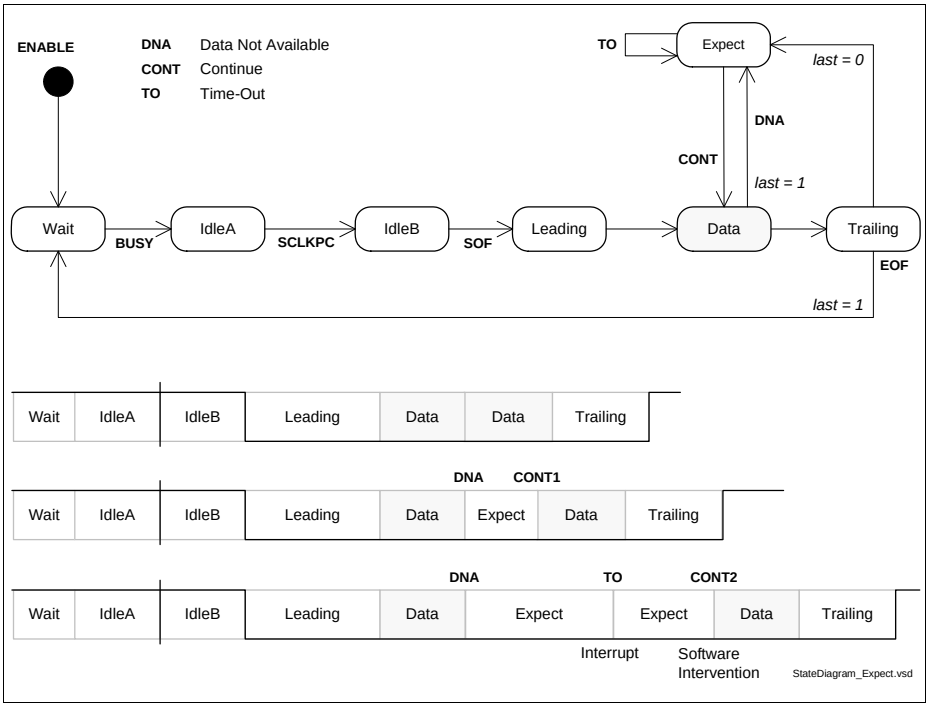


Figure 20-21 EXPECT Cycles of the QSPI State Machine

20.5.7 External Slave Select Expansion

In this mode the bit field **BACON**.CS drives directly the SLSO1 to SLSO4 signals, not demultiplexed by the QSPI, but chip-externally. **SSOC**.AOL bits can invert the SLSO0...4 signal levels individually; **SSOC**.OEN bits enable them. See also **GLOBALCON**.DEL0.

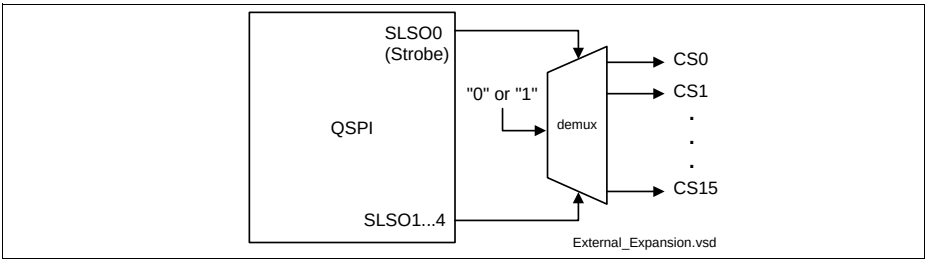


Figure 20-22 Demultiplexing the Slave Select Signals Externally

In order to ensure glitch free selection, a strobe signal is provided, driven at SLSO0 pin. This signal is delayed relative to the SLSO1...4 signals for LS (Lead_Strobe) and TS (Trail_Strobe) delays, equal in duration and configurable in the range of 1 to 16* f_{BAUD2}. The duration of the LS and TS delays is set in the **GLOBALCON**.STROBE.

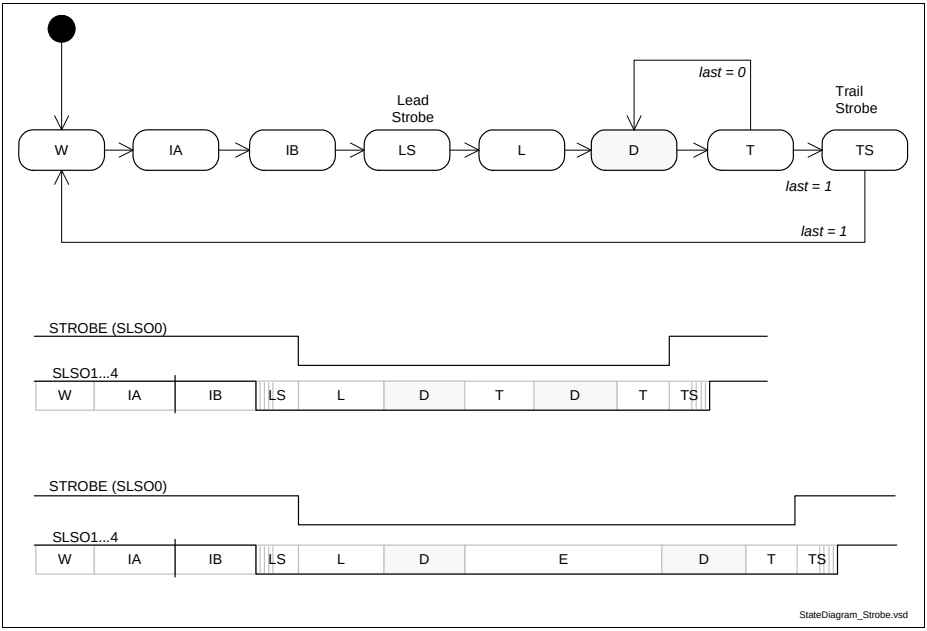


Figure 20-23 State Machine and SLS Signals in Strobed Mode

20.6 User Interface

This section describes the possibilities available for transferring data between the on-chip RAM memories and the QSPI module using the transmit and receive FIFOs. The QSPI features one 4 x 32-bit Tx FIFO and one 4 x 32-bit Rx FIFO. The address ranges for writing data and configuration to the Tx FIFO and for reading data and configuration from the Rx FIFO consist of address locations with special properties:

- Tx FIFO
  - DATA_ENTRY - writes to any of these eight functionally identical locations are always interpreted as data
  - BACON_ENTRY - writes to this location are always interpreted as configuration
  - MIX_ENTRY - writes to this location are interpreted as data or configuration, based on a set of rules
- Rx FIFO
  - RX_EXIT - reads from this location deliver either data or data and status, based on a set of rules.

20.6.1 Transmit and Receive FIFOs

The Tx_FIFO is filled with both configuration and data entries, which are distributed to the **BACON** and the shift register at the exit of the Tx_FIFO. The TxFIFO keeps track if an entry is configuration or data. The control and the data elements are automatically distributed to the **BACON** and shift register. If no data is available, the new BACON is pending.

If a frame is finished and there is data in the FIFO

- in short mode, a new frame starts automatically with the same configuration as the last one.
- in long mode, the extra data are ignored.

TXFIFO error conditions:

- FPI bus write to a full TXFIFO generates an overflow interrupt. The write is ignored.
- In case of slave mode, (**GLOBALCON**.MS = 1X), in case of an underflow, only “1” are delivered.

RXFIFO error conditions:

- FPI bus read from an empty RXFIFO generates an underflow interrupt, and delivers only “1” bits.
- Hardware attempt to write a full RXFIFO with data or status generates an overflow interrupt. The write attempt is ignored by the RXFIFO.
- If the **GLOBALCON**.SRF bit is set, the communication is stopped in case of full FIFO in master mode. In slave mode, the bit is ignored, no stop can be done.
- If the FPI bus frequency is very slow, and the baud rate very high, it can happen that some data is lost on the way between the shift register and a not full RXFIFO. In such a case, an overflow interrupt is generated, although the RXFIFO is not full.

The Rx_FIFO can be filled with both status and received data entries, depending on **GLOBALCON**.SI.

Note: If a 8-bit or 16-bit write is executed to the TXFIFO entry addresses, the data is mapped 1:1 to its location within the 32-bit wide TXFIFO entry, and the rest is padded with zeros.

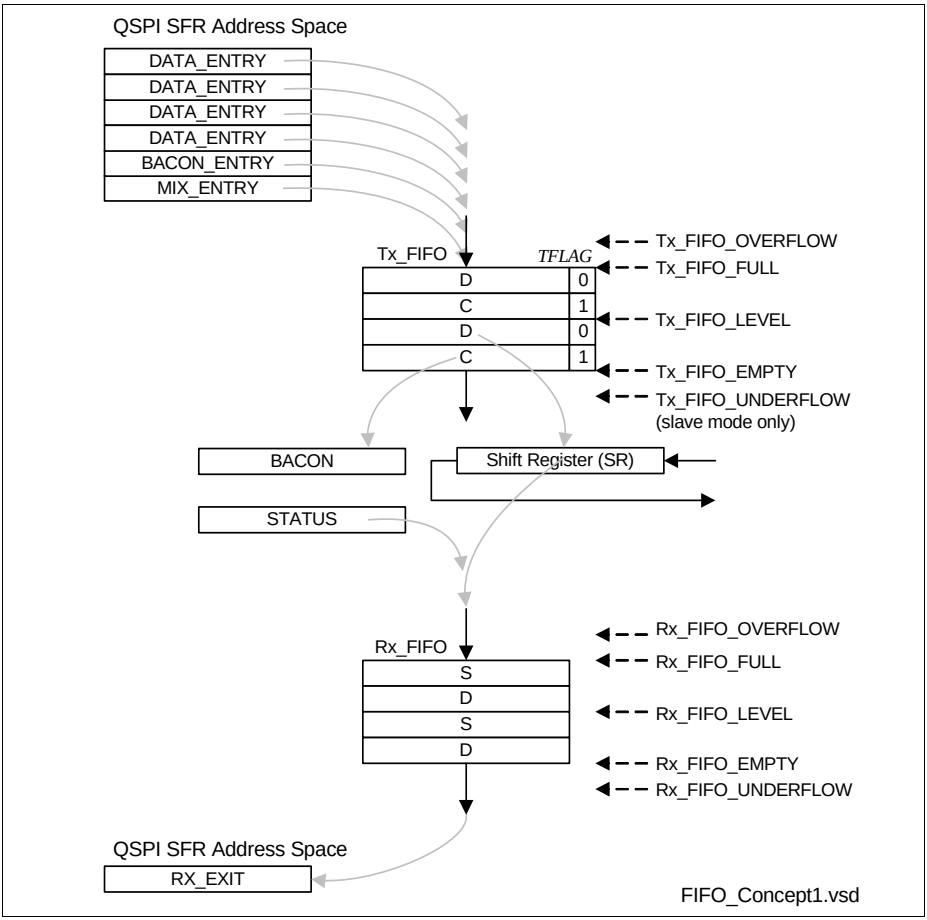


Figure 20-24 Architecture of the Tx and Rx FIFOs

(>>**Interrupts**)

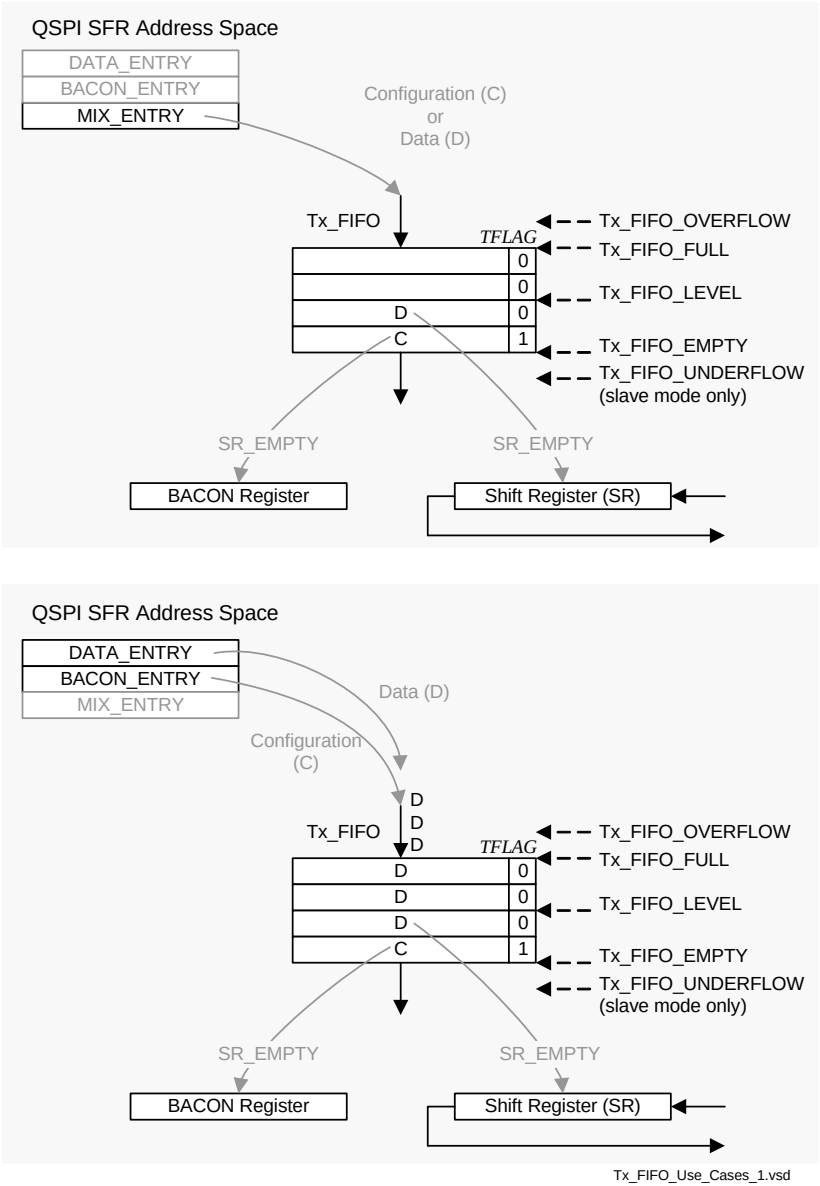


Figure 20-25 Using Tx FIFO Mix and Bacon / Data Entries

20.6.1.1 Short Data Mode

In Short Data Mode, the QSPI module transmits single data with a length of 2 to 32 bits in one frame. This mode is defined by **BACON**.LAST = 1 and **BACON**.BYTE = 0.

The transfer cycle is a sequence of the following phases: W?_I_L_D_T. The symbol “?” means that the phase is optional (duration of 0 allowed).

For example, a Tx_FIFO event can be used to trigger one DMA transfer (consisting of two 32-bit moves) to transfer both the BACON and DATA entries from an on-chip RAM memory to the QSPI module.

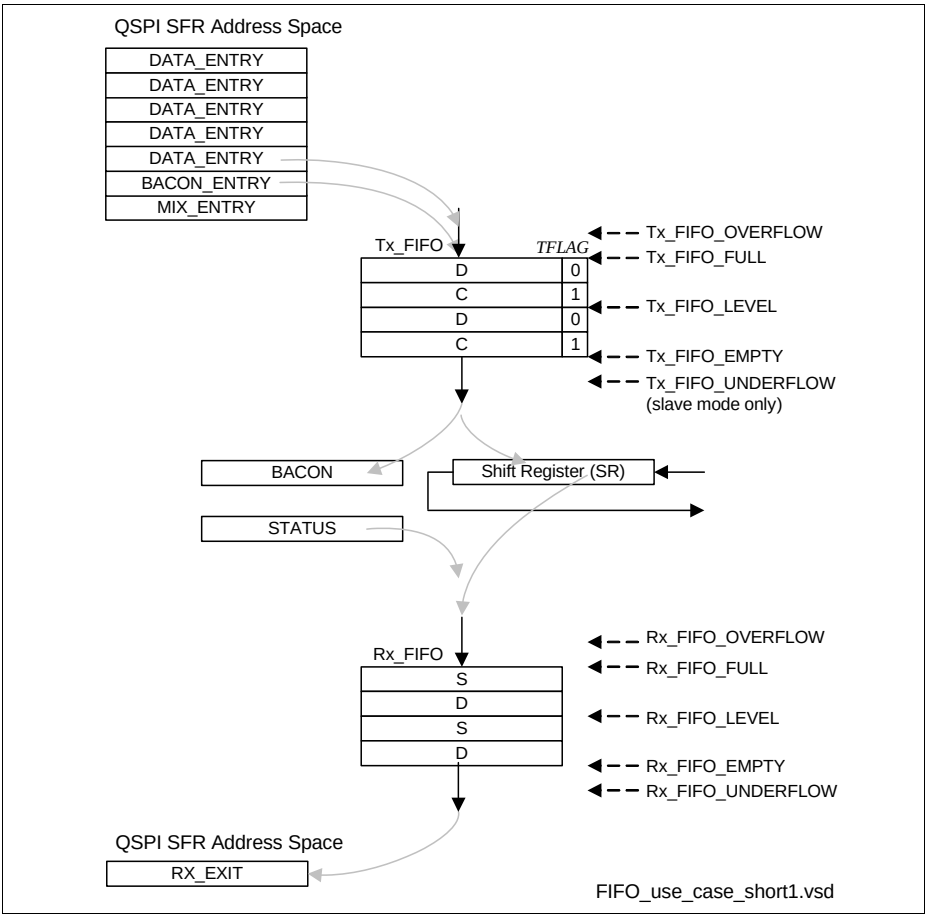


Figure 20-26 User Interface in Short Data Mode

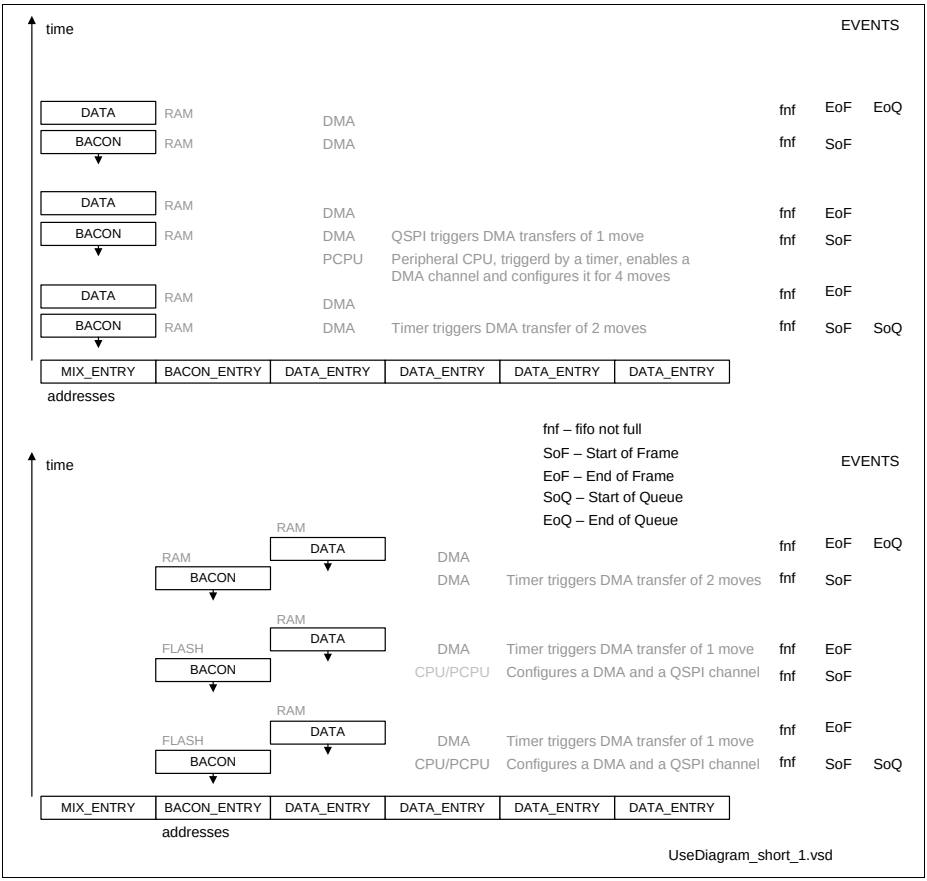


Figure 20-27 Use Diagram of the User Interface in Short Data Mode

20.6.1.2 Long Data Mode

In Long Data Mode, the QSPI module transmits bursts of up to 32 bytes (256 bits) in one frame. This mode is defined by programming **BACON**.LAST = 1 and **BACON**.BYTE = 1.

One transfer cycle in long data mode is a sequence of the following phases: W?_I_L_D_T. The W_I_L_D_T_I_L_D_T sequence is possible. The symbol “?” indicates that the phase is optional (duration of 0 allowed). Each “D” indicates a data length as defined with **BACON**.DL and **BACON**.BYTE, that is, up to 256 bits. One “D” is made of a number of 32-bit TXFIFO entries “d” and one rest at the end “b”, which can be written as D=ddddddb.

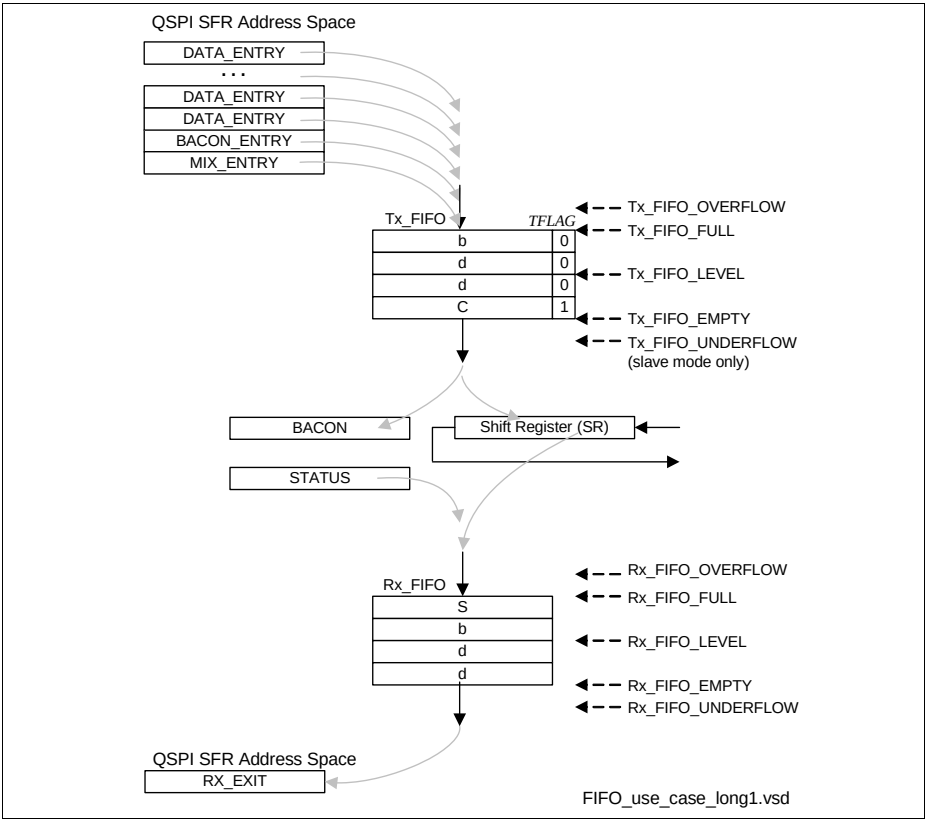


Figure 20-28 User Interface in Long Data Mode

In a DMA transfer example, a Tx_FIFO_EMPTY event can be used to trigger one DMA transfer (consisting of, for example, four 32-bit moves) to transfer the BACON and three DATA entries from RAM memory to the QSPI module. The three data words could

contain 80 bits (10 bytes) of data. The three data words could also contain 40 bits (5 bytes), 64 bits (8 bytes) plus one dummy word. In this way short bursts of up to 96 bits can be transmitted.

In such 4 words DMA transfer example, the Tx_FIFO event can be used to transfer two 32-bit words per transfer.

If a configuration word is written when a data word is expected, the configuration word is ignored, and an Unexpected Configuration Error is raised.

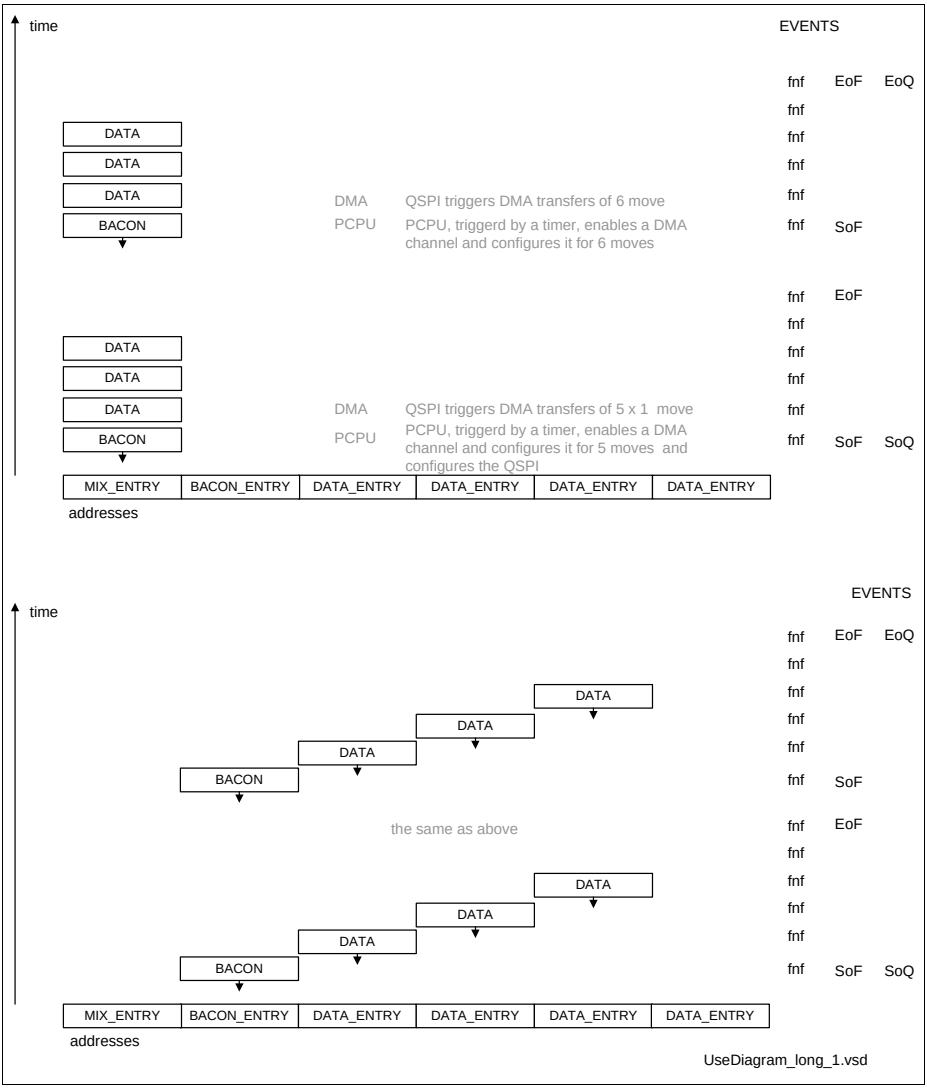


Figure 20-29 Use Diagram of the User Interface in Long Data Mode

20.6.1.3 Continuous Data Mode

In Continuous Data Mode, the QSPI transmits a stream of data with an arbitrary length with an active slave select signal. It can be used with both short and long data.

This mode starts by programming a control word containing **BACON**.LAST = 0 and the first data. The communication continues with the subsequent data entries written. The mode ends with a control word containing **BACON**.LAST = 1 and the last data. If LAST = 0, then TRAIL is a delay between data blocks. The W_I_L_D_T_D_T_D_T sequence appears. Each “D” represents data block as defined with **BACON**.DL and **BACON**.BYTE (up to 256 bits). If LAST = 1 then TRAIL is trailing delay, see **Figure 20-9**.

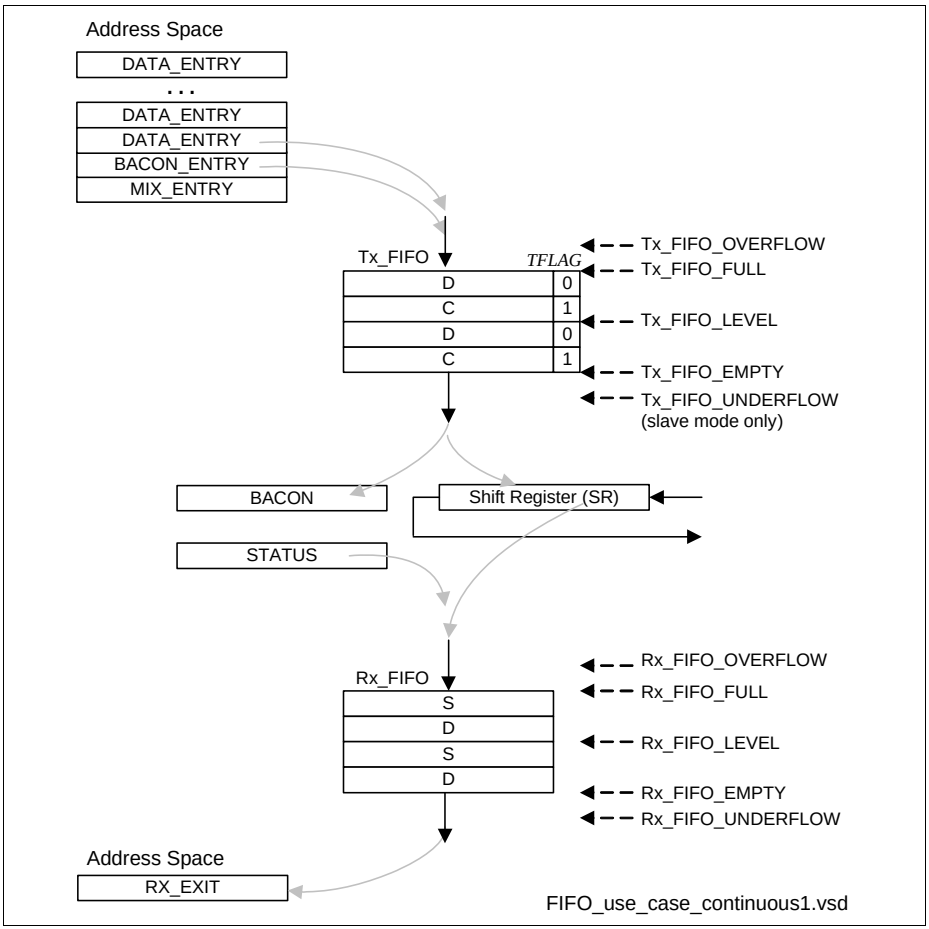


Figure 20-30 User Interface in Continuous Data Mode

The **Figure 20-31** shows an example of continuous data mode using short data. Here, the CPU starts the stream by writing BACON to the BACON_ENTRY address, and afterwards a DMA transfers the whole stream except the last BACON and DATA.

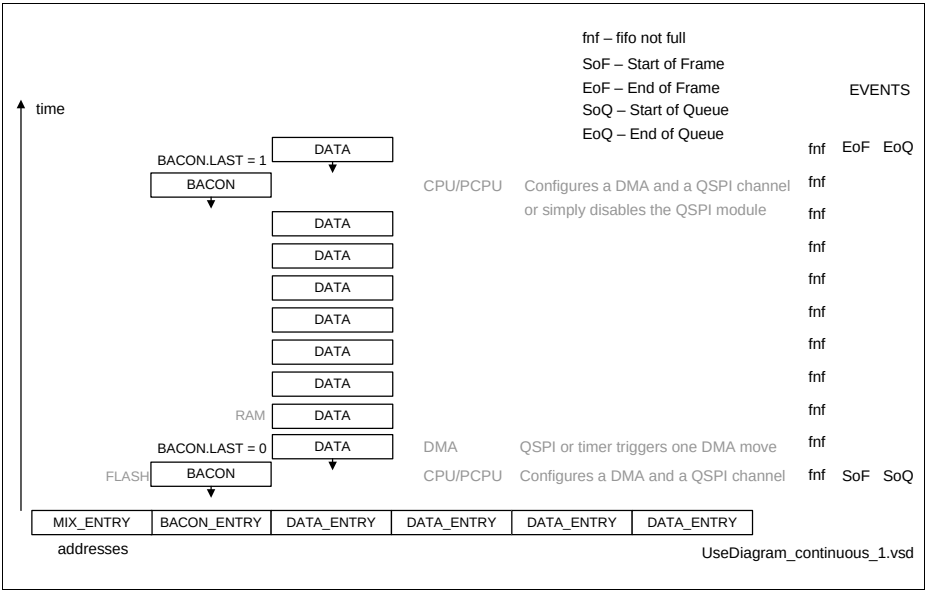


Figure 20-31 Use Diagram of Continuous Data Mode with Short Data

The **Figure 20-32** shows an example of continuous data mode using long data. Here, the first BACON must be repeated after each data block, but on the other hand, the whole transfer can be done using only the MIX entry and one single DMA channel, without CPU involvement. The loss of efficiency because of the unnecessary extra BACONS can be minimized by using long data blocks.

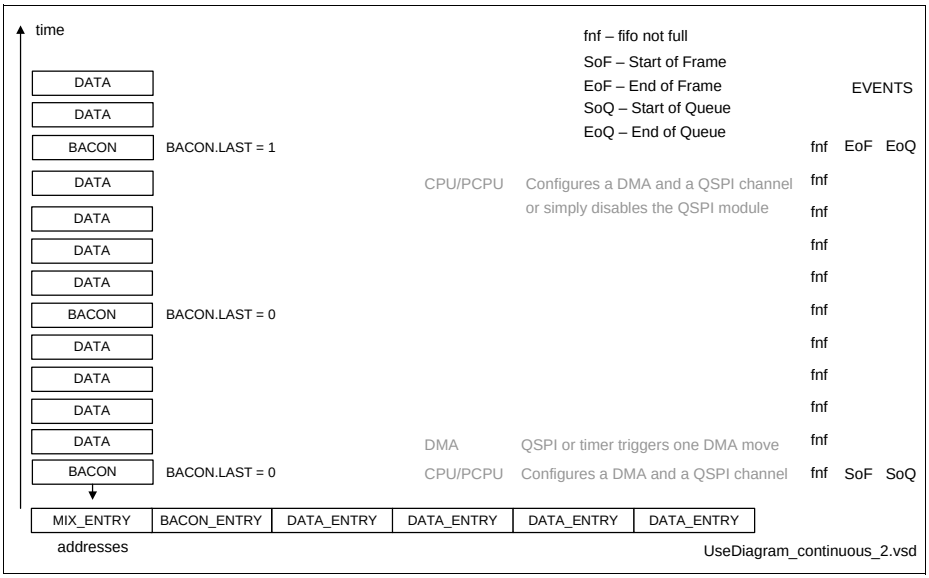


Figure 20-32 Use Diagram of Continuous Data Mode with Long Data

20.6.1.4 Single Configuration - Multiple Frames Behavior

If longer sequences of data with length of up to 16 bit have to be sent to single slave in separate frames, then the DMA move for the repeating configuration word can be saved. An example of a TxFIFO contents and the resulting message sequence is shown in **Figure 20-33**.

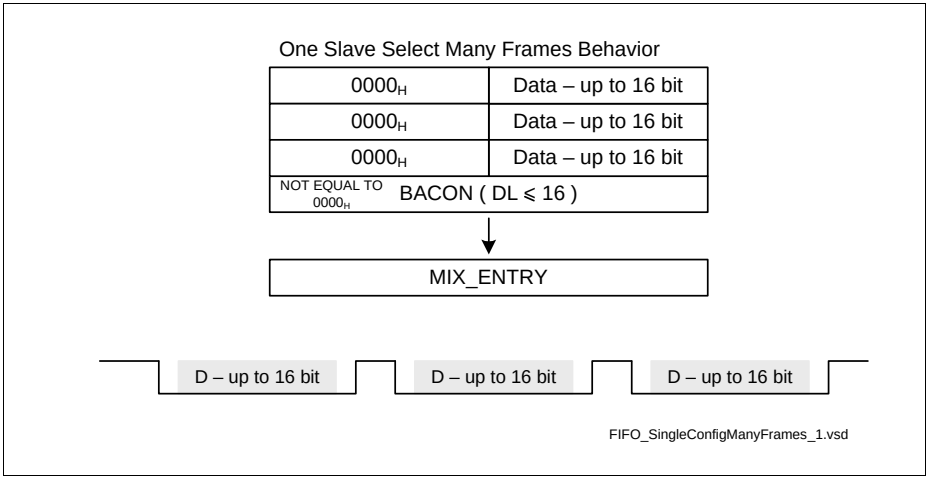


Figure 20-33 Multiple Frames to One Channel (Example of up to 16 Bit Data)

In order to support sending short multiple frames to single channel without having to send the same configuration each time, the TXFIFO follows this rule:

- If **BACON**.DL less or equal 15 (up to 16 bit data length)
- if MIX_ENTRY address is used
- then if the upper 16 bit of a 32-bit MIX_ENTRY are 0, the entry is considered data, else configuration. Note that there is no valid configuration word with upper 16 bits equal to 0.
- In such a case the TXFIFO marks the entry as data.

*Note: In case the **BACON**.LAST = 0, the slave select remains active between the frames. The behavior is equal to the continuous mode. The disadvantage of this method is that only up to 16 bits data can be transferred with one FPI bus move. The advantage is that continuous mode is possible using the mix entry.*

20.6.1.5 Big Endian Data Format

The bit-field **ECON**.BE activates the permutation of the data bytes written to the shift register and read from it to the big endian format and back.

ECON.BE allows for switching between big and little endian per slave (SLSO signal).

See **Figure 20-34** and **Figure 20-35**.

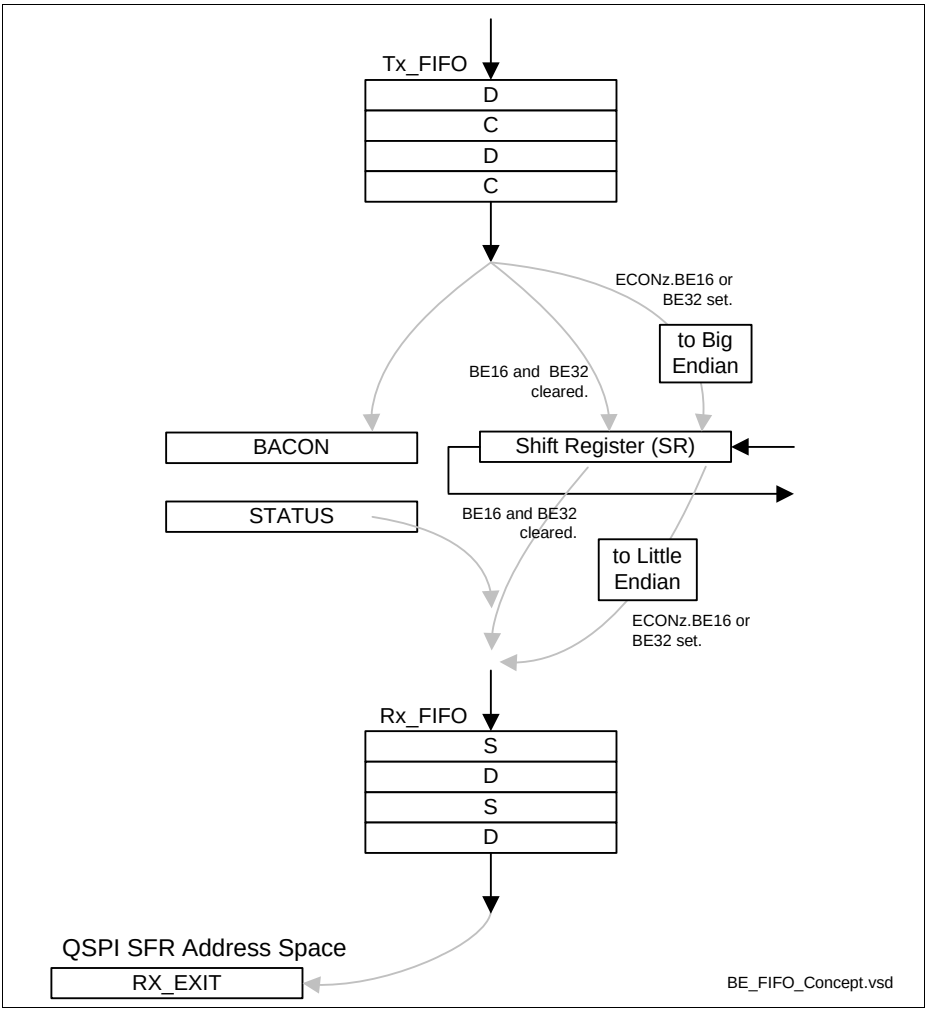


Figure 20-34 Big Endian Data Path

Converting data from little to big endian and other way around requires one and the same permutation of the data bytes, as shown in **Figure 20-34**, One permutation block is located before and one after the shift register, so that only data can be permutated, and not **BACON** or **STATUS**.

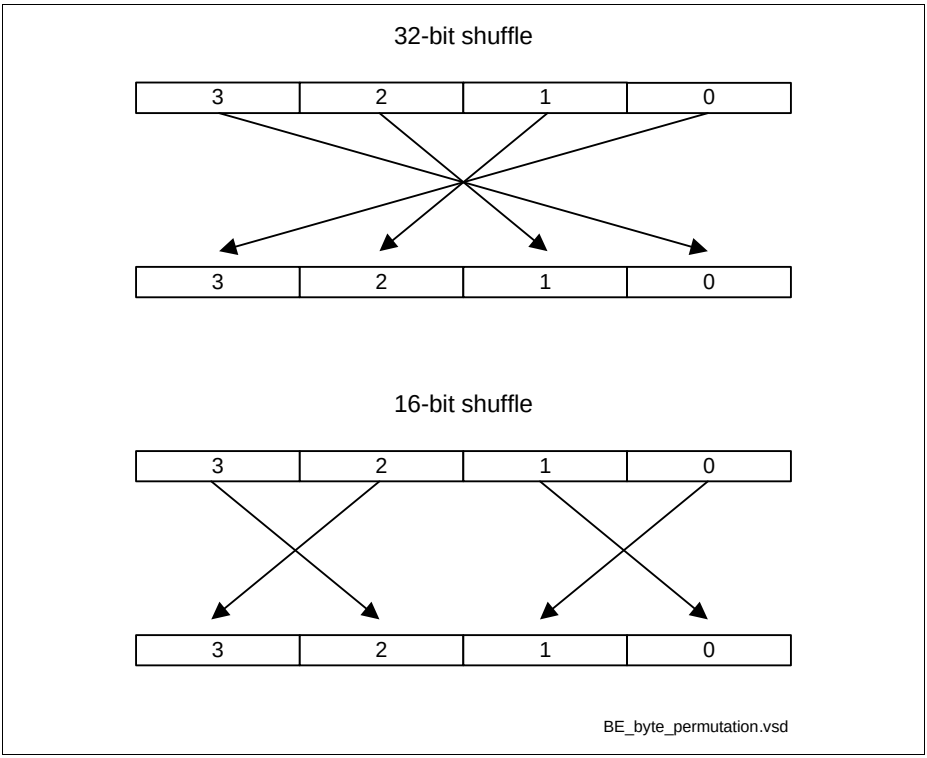


Figure 20-35 Endianness Byte Permutation

The endianness byte permutation should be used only for 16-bit and 32-bit data (and multiples of it), and not for other data lengths.

20.6.2 Loop-Back Mode

Loop Back mode is a master mode test feature used for establishing a module internal connection between the transmit and the receive signal, the same as if the transmit and receive pins would be short connected. It is actived by setting the **GLOBALCON**.LB bit.

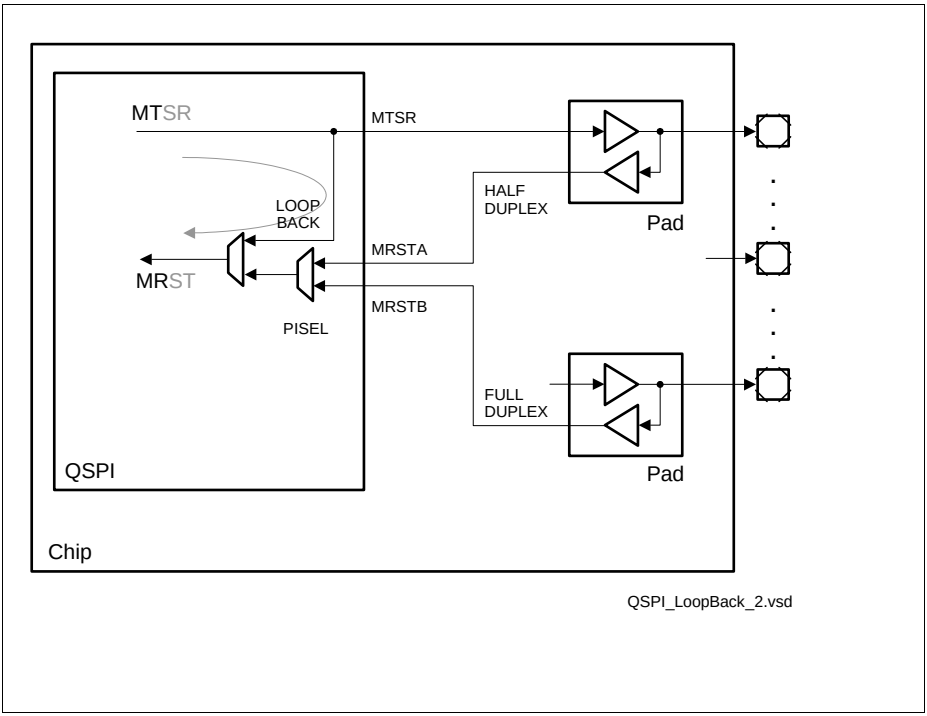


Figure 20-36 Loop-Back Mode

20.7 Interrupts

There are several types of interrupts:

- Interrupts related to the user interface
  - FIFO related interrupts
- Interrupts related to the shift engine (state machine)
  - error interrupts
  - phase transition interrupts

According to this classification, here follows the complete list of all the interrupts:

- FIFO related interrupts
  - TX - Transmit FIFO Interrupt, requests feeding of the TXFIFO, but does not signal the error events of overflow and underflow (see **Figure 20-24**)
  - RX - Receive FIFO Interrupt, requests emptying of the RxFIFO , but does not signal the error events of overflow and underflow (see **Figure 20-24**)
- ERR Error Interrupt
  - signals every error condition
  - signals TX_FIFO_OVERFLOW, TX_FIFO_UNDERFLOW
  - signals RX_FIFO_OVERFLOW, RX_FIFO_UNDERFLOW
- PT - Phase transition interrupt, signalling two out of all phase transitions in the QSPI communication cycle (see **Figure 20-18** and **Figure 20-21**)
  - PT1 event
  - PT2 event
- U - User Interrupt
  - triggered by the PT1 event during the time **BACON**.UINT bit keeps this event enabled.

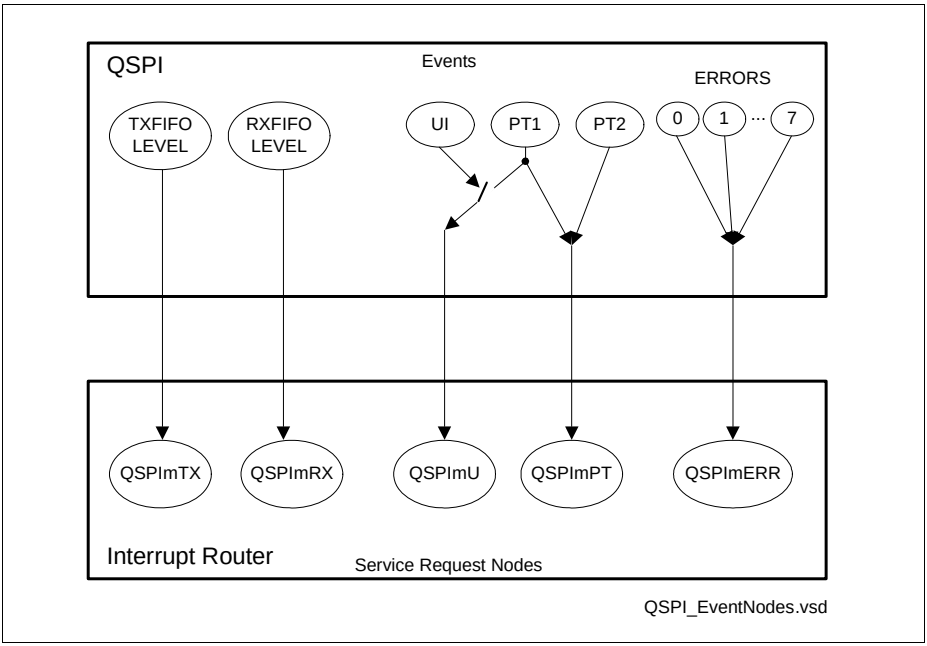


Figure 20-37 Events and Interrupts Overview

20.7.1 Slave Mode SLSI Interrupt

In slave mode, deactivating the SLSI signal (SLSI rising edge) triggers the PT interrupt, if PT2 enabled.

20.7.2 Interrupt Flags Behavior

An state of an event flag (low or high) does not influence the flow of its event signal. If an event occures recurrently, and the event flag remains set (if not cleared by the software interrupt handler), the interrupt flag (which is cleared by hardware, by an ICU, when the interrupt wins an arbitration round) would receive recurrent triggers, and the flag would be constantly set all the time.

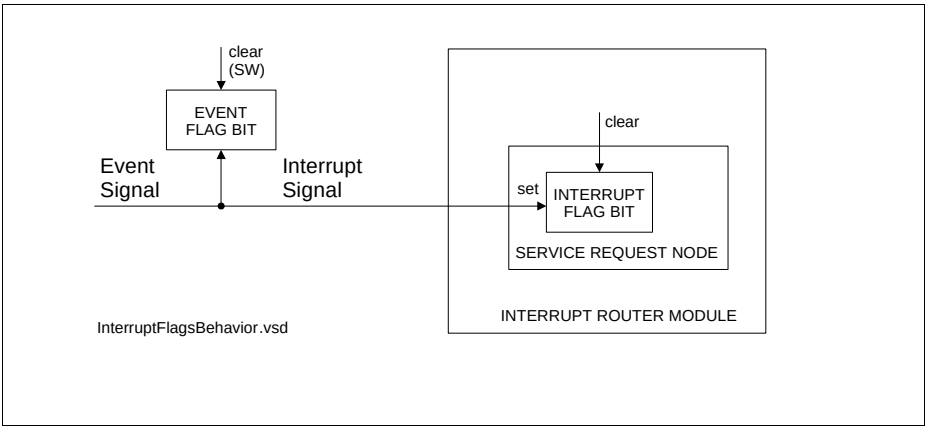


Figure 20-38 Events and Interrupts Overview

20.7.3 TXFIFO Interrupt Generation

The TXFIFO provides two interrupt generation modes, selected by the bit-field **GLOBALCON1**.TXFM:

- Single Move Mode
- Batch Move Mode
- Combined Mode

The RXFIFO and the TXFIFO can use the interrupt generation modes independently of each other.

Single Move Mode

The purpose of the Single Move Mode is to keep the TXFIFO as full as possible, refilling the TXFIFO by writing to it as soon as there is a free element.

The single move mode supports primarily a DMA operation using single move per TXFIFO interrupt. In this mode the DMA keeps the TXFIFO full. A DMA request is triggered each time a write to the TXFIFO is performed, and there is at least one empty element available. If the write makes the TXFIFO full (no more empty elements), an interrupt is not generated in order to prevent overflow. It is generated later when the shift register (or BACON) reads one element from the TXFIFO, making it not full.

The **Figure 20-39**, shows the filling levels of the TXFIFO, and the events associated with each filling level triggering a TXFIFO refill interrupt.

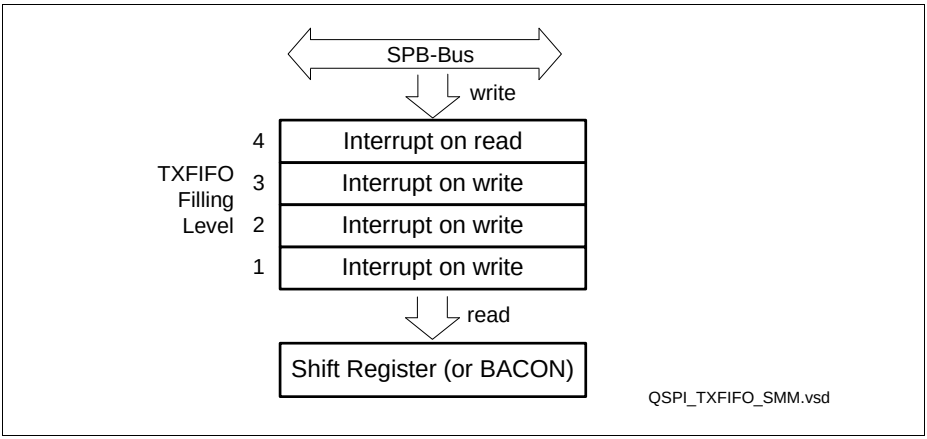


Figure 20-39 Interrupt Generation in the Single Move Mode

In order to initiate the very first chain of the refill interrupts after power-on reset, it is necessary that the software either performs one write to the TXFIFO, or sets the interrupt flag in the TXFIFO interrupt node. Afterwards, the (DMA) interrupt-chain is self sustaining until the whole transaction is over.

Attention: In Single Move Mode multiple software writes or block DMA moves would lead to multiple interrupts and (false) transaction lost events. Therefore they should be avoided - only single moves should be used.

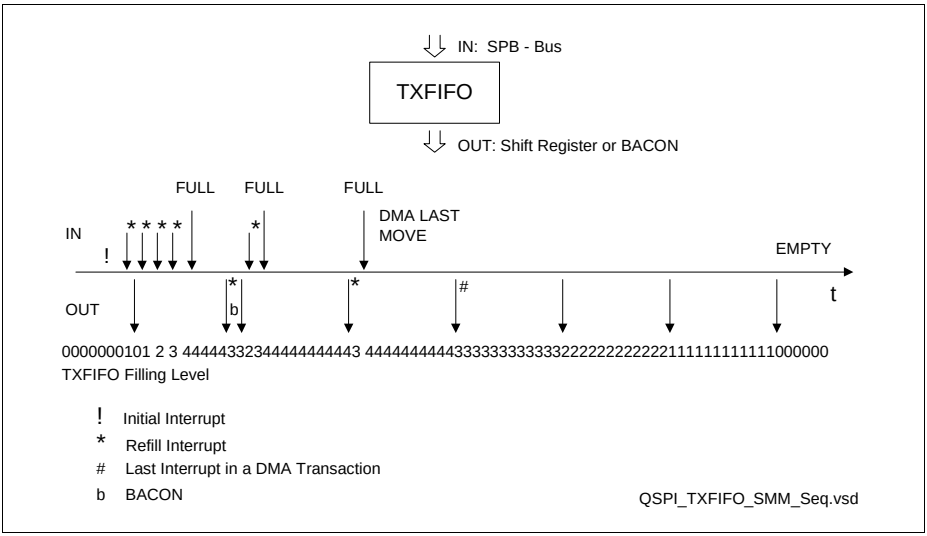


Figure 20-40 TXFIFO - Interrupt Operation in Single Move Mode

Batch Move Mode

Batch Move Mode supports the following use case:

- CPU servicing the TXFIFO

The purpose of the Batch Move Mode is to reduce the number of interrupts by triggering an interrupt when more then one TXFIFO elements are empty. For example, a CPU can be interrupted less frequently and it can perform more than one moves per interrupt.

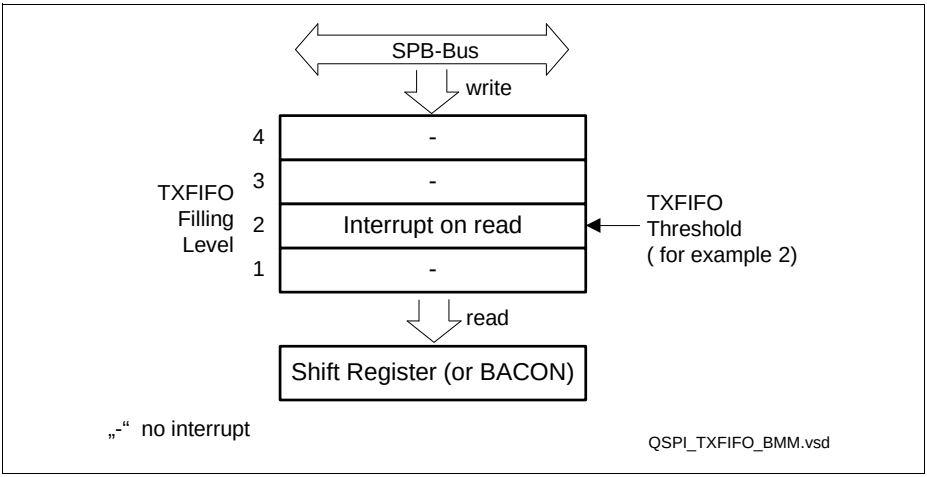


Figure 20-41 TXFIFO - Interrupt Generation in the Batch Move Mode

In Batch Move Mode, an interrupt is generated only at one point in the TXFIFO, when the filling level falls below the programmed threshold, implying that there are at least the predefined number of empty TXFIFO elements available.

Attention: The TXFIFO must be refilled until the filling level has risen above the interrupt threshold (ensured by polling the filling level), in order to guarantee that the next interrupt will occur.

At high baud rates and short frames, it could be possible that a couple of frames have been transmitted until the moment the TXFIFO is refilled. For example, if the threshold is set to three full elements, then the interrupt will be set when the filling level falls below three, making two elements free. If the refill moment of the TXFIFO is delayed so long that two frames have been transmitted (or frame and BACON), the TXFIFO will become empty, and refill sequence of two will not reach the level of three. So, the next interrupt would never come unless the filling level is being polled and the CPU keeps refilling in a loop until the threshold has been passed (ideally the TXFIFO would be filled completely with each batch of moves).

Combined Mode

TXFIFO generates interrupt each time a data is fetched from it below the pre-programmed threshold, as defined in the **GLOBALCON1**.TXFIFOINT. Consequently, it generates two close interrupts when delivering BACON and the first following data. If the BACON TX interrupt is not serviced until the DATA1 TX interrupt comes, which will always be the case due to their time proximity, the BACON TX interrupt will be lost. Therefore, it is recommended that a DMA should always perform two moves per interrupt trigger (transfer size of two). Otherwise, the TXFIFO average filling level will go down with the time and at the end remain oscillating between zero and one (except in continuous mode). The DMA transaction loss event that may be raised by the DMA should be ignored.

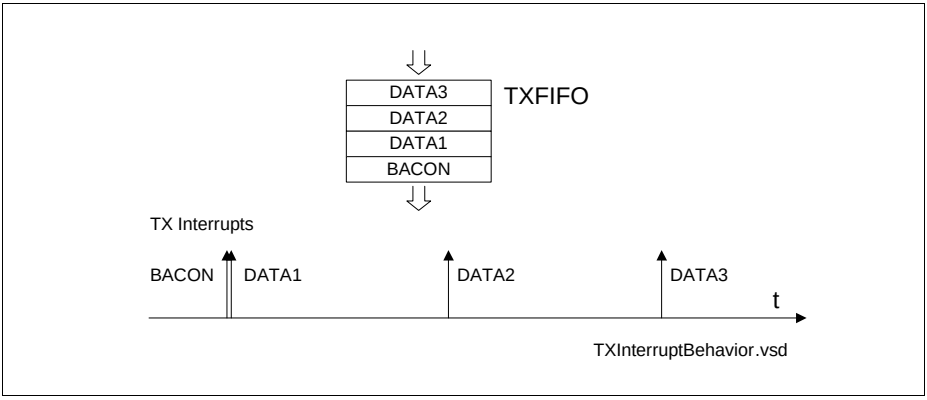


Figure 20-42 Events and Interrupts Overview

20.7.4 RXFIFO Interrupt Generation

The RXFIFO provides two interrupt generation modes, selected by the bit field **GLOBALCON1**.RXFM:

- Single Move Mode
- Batch Move Mode
- Combined Mode

Single Move Mode

The purpose of the Single Move Mode is to keep the RXFIFO as empty as possible, by fetching the received elements one by one as soon as possible.

The single move mode supports primarily a DMA operation using single move per RXFIFO interrupt. In this mode the DMA keeps the RXFIFO as empty as possible. A DMA request is triggered each time a read from the RXFIFO is performed, and the RXFIFO is afterwards still not empty. If the read makes the RXFIFO empty, an interrupt is not generated in order to prevent underflow. It is generated later when the shift register (or STATUS) writes new element to the empty RXFIFO.

The **Figure 20-39**, shows the filling levels of the RXFIFO, and the events associated with each filling level triggering a RXFIFO refill interrupt.

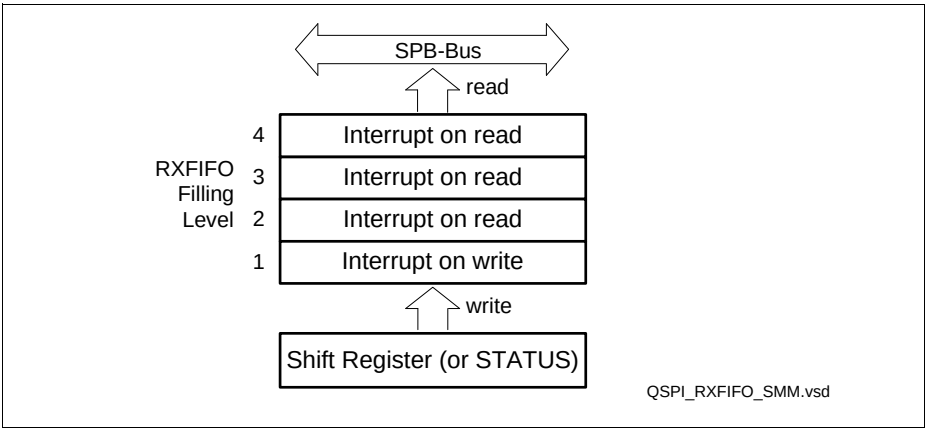


Figure 20-43 RXFIFO - Interrupt Triggering in the Single Move Mode

The initial RXFIFO interrupt is triggered by the Shift Register, after it delivers the first received element. Afterwards, the DMA trigger-chain of refetch interrupts is self sustaining until the whole transaction is over. At the end of the DMA transaction, there is no service request remaining active in the service request node, due to the fact that the read of the last element in the RXFIFO does not trigger an interrupt.

Attention: In Single Move Mode multiple software reads or block DMA moves lead to multiple interrupts and (false) transaction lost events. Therefore they should be avoided - only single moves should be used.

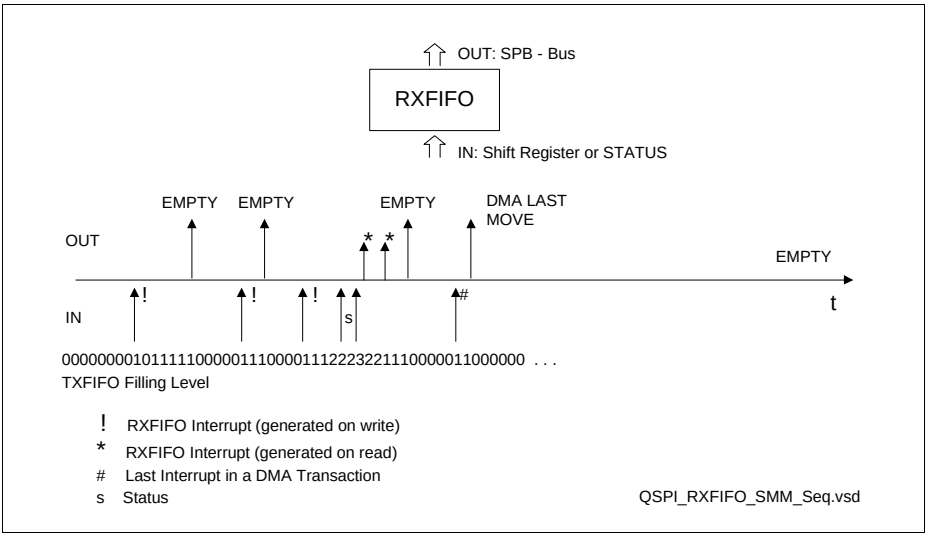


Figure 20-44 RXFIFO - Interrupt Operation in Single Move Mode

Batch Move Mode

The purpose of the Batch Move Mode is to reduce the number of interrupts by triggering an interrupt when more then one RXFIFO elements are full. For example, a CPU can be interrupted less frequently and it can perform more than one move per interrupt.

Batch Move Mode supports the following use case:

- CPU servicing the RXFIFO

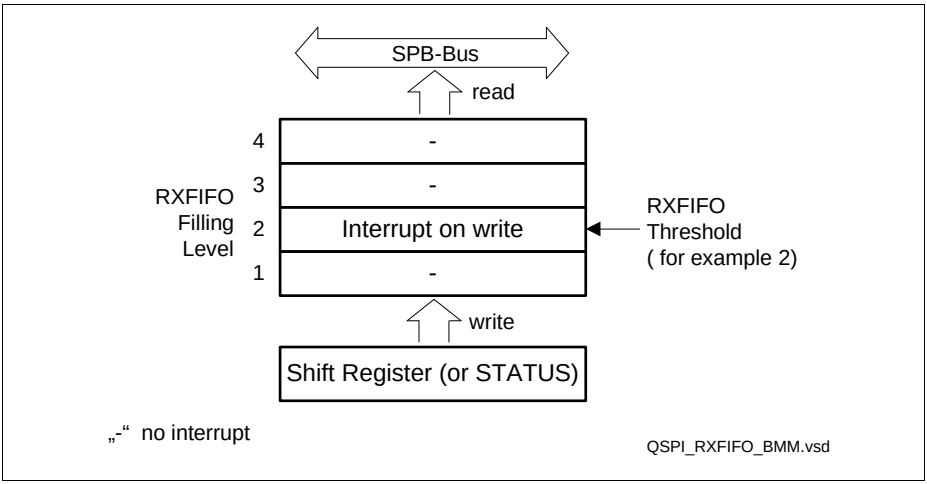


Figure 20-45 RXFIFO - Interrupt Triggering in the Batch Move Mode

In Batch Move Mode, an interrupt is generated only at one point in the RXFIFO, when the filling level rises above the programmed threshold, implying that there are at least the predefined number of full RXFIFO elements available.

Attention: It must be guaranteed that the CPU keeps emptying the RXFIFO and polling the RXFIFO filling level until the filling level has fallen below the interrupt threshold (or the RXFIFO is empty), in order to guarantee that next interrupt will occur.

At high baud rates and short frames, it could be possible that more than one frames have been received until the moment the RXFIFO is reemptied. For example, if the threshold is set to two full elements, then the interrupt will be set when the filling level rises above two. If the emptying moment of the RXFIFO is delayed so long that two additional frames has been received (or a frame and STATUS), the RXFIFO will become full, and reempty sequence of two will not reach the level of two. So, the next interrupt would never come. This effect can not occur with threshold levels of 3 and 4. The threshold level of 1 is possible, but does not make much sense (use single move mode instead).

Combined Mode

RXFIFO generates interrupt each time a data is written to it above the preprogrammed level, as defined in the **GLOBALCON1**.RXFIFOINT. Consequently, it generates two close interrupts when filled with data and STATUS. If the data RX interrupt is not serviced until the STATUS RX interrupt comes, which will always be the case due to their time proximity, the STATUS RX interrupt will be lost. In such a case, it is recommended that the DMA performs two moves per interrupt trigger (transfer size of two). Otherwise, the RXFIFO average filling level will go up with the time and at the end overflow.

20.7.5 DMA Transfer Example

Transmitting one compact RAM queue using DMA could consist of the following steps:

- The user software configures one appropriate DMA channel (source address, destination address, one move per transfer, number of transfers per transaction, gating mode, single mode, interrupt at the end of the transaction etc...)
- A not full TxFIFO from some QSPI module constantly generates pending DMA request
- A timer triggers a software routine that enables the DMA channel
- The DMA channel reads the queue elements from some on-chip general purpose RAM and writes them to the QSPI, one element at a time. One DMA transfer is one move long. One whole queue takes one transaction.
- The DMA channel generates an interrupt at the end of the transaction
- The DMA channel disables itself at the end of the transaction (single mode) and waits for further software actions.

The sequence is identical for the receive direction.

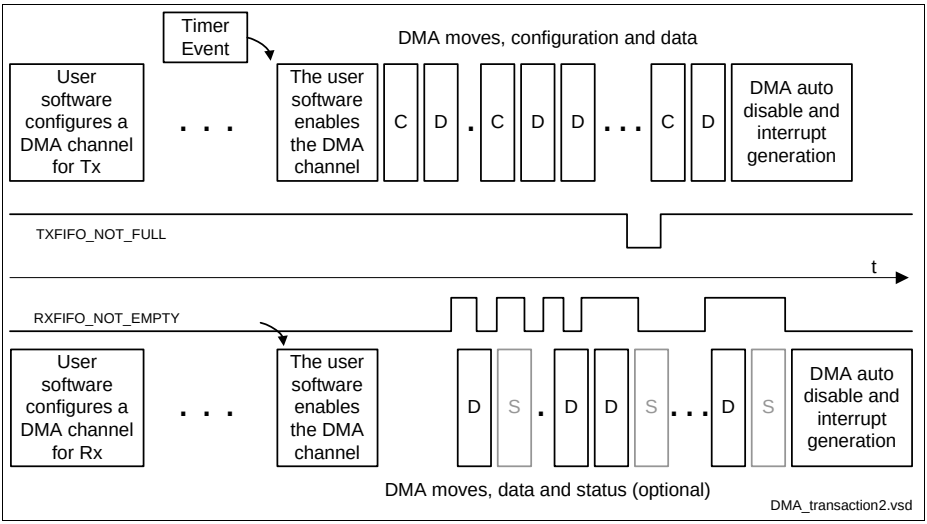


Figure 20-46 Example of transmitting a compact RAM Queue per DMA

20.8 Slave Mode

The slave mode is entered by setting the bit field **GLOBALCON**.MS to value 1x. Before entering the slave mode, the related parameters must be configured, including pin selection for SCLKI, SLSI and MTSR, pin input level, and pull-up/pull-down configuration. Subsequently state machine reset should be performed and afterwards the slave mode should be entered.

In slave mode data is being pushed into or out of the module through the serial pins by an external master. The RXFIFO must be regularly emptied by the system in order to avoid data loss, and the TXFIFO must be regularly fed by the system to avoid delivering all “1” data in case of empty TXFIFO.

If the slave mode relies on a slave select signal to mark the start and, in parallel to the bit counting, the end of a frame, then deactivating the slave signal by the master automatically resets the shift register state machine to wait state.

If the slave mode relies solely on bit counting for determining the frame end, then any SCLKI clock period longer than two bit times is treated as an end of a frame and simultaneously as a baud rate error.

In slave mode, the module simply immediately responds to external clock edges. Therefore, the settings for the leading, trailing, idle delays, and duty cycle and sampling point are irrelevant. From all timing settings, only the data length and the baud rate divider setting are relevant. The baud rate setting is important because of its role of watchdog timer of the incoming serial clock.

Generally, the slave mode supports the same user interface as the master mode. This means the BACON-Data sequences remain the same. The SRF bit is ignored.

Limit the data length setting in the slave mode to the range of 2 to 32 bits. The byte setting **BACON**.BYTE is to be set to zero, that is, it always defines bits.

In case of receive only (simplex receive) the user software must reset the module in order to switch to transmit and receive mode (full duplex). Generally, mode reconfiguration (slave mode to master mode) requires module reset which resets the internal state machines and counters.

In case of slave transmit, the user software must write data to the TXFIFO before the first SCLKI edge of a frame arrives. If the TXFIFO is empty at this point of time, an underflow occures, all “1” data is delivered, and underflow error interrupt is triggered if enabled. If the TXFIFO is written at the same time when the first SCLKI edge starts the shifting, the underflow occures, interrupt will be triggered and data from the TXFIFO will be transmitted possibly corrupted - the first bit will be “1”.

In case of TXFIFO / RXFIFO underflow the fifos deliver all “1” data .

In case of TXFIFO / RXFIFO overflow, the last data is lost. In such case data or optionally status in the RXFIFO can be lost.

The bit counting in the slave mode operates according to the following rules:

- The incoming bits (or shift clock shift edges) are counted and when the number defined in the **BACON**.DL is reached, the bits are transferred in the RXFIFO
- If the SLSI input is deactivated, the internal bit counter is reseted.

In slave mode, sleep requests may be enabled only when there is no pending transmit queue and no pending transmission and the TXFIFO is empty.

20.8.1 Shift Clock Phase and Polarity in Slave Mode

In slave mode the shift clock phase and polarity are fixed to CPH=1 and CPOL=0 (have to be programmed accordingly in the **ECON** register selected by **BACON**.CS) and slave select polarity is fixed to low active. The baud rate that the slave expects is defined in the **ECON** register pointed at by the **BACON**.CS.

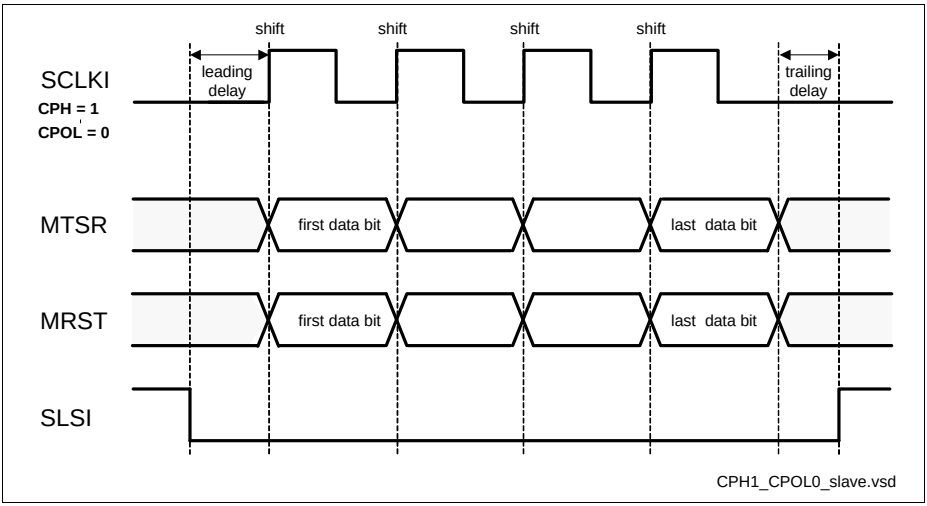


Figure 20-47 Slave transfer, CPH = 1, CPOL = 0

20.8.2 Shift Clock Monitoring

The QSPI module provides two mechanisms for monitoring the shift clock input signal in slave mode. These mechanisms monitor:

- if the receiving shift clock frequency (baud rate) lies within a certain range and
- if there are spikes on the shift clock input.

In case such disturbances are detected, the reaction is always the same:

- the corresponding error flags in **STATUS**.ERRORFLAGS are set, which can be cleared with **FLAGSCLEAR**.ERRORCLEARS
- the common error interrupt is raised if enabled in **GLOBALCON1**.ERRORENS and
- the bit **GLOBALCON**.EN is automatically cleared, if this feature is enabled by the bit **GLOBALCON**.AREN. The user software can activate a reset afterwards

In case of too high baud rate error , baud rate detection and spike detection both react, where spike detection is more effective. Spike detection reacts immediately, baud rate detection requires several bits with higher baud rate. Baud rate detection uses a window two nominal bit times wide and counts the real bit times: double baud rate error requires at least four real shift clock pulses to be detected, triple baud rate error requires six and so on. Baud rate detection does not react to all spikes smaller than one kernel clock period.

In case of too low baud rate error, baud rate detection reacts immediately, spike detection ignores this case.

20.8.2.1 Baud Rate Error Detection

If the shift clock has less than half or more then double the expected baud rate, then:

- the corresponding flag is set in **STATUS**.ERRORFLAGS
- error interrupt is raised if enabled in **GLOBALCON1**.ERRORENS
- automatic clear of **GLOBALCON**.EN follows, if enabled by **GLOBALCON**.AREN.

20.8.2.2 Spike Detection

Short spikes on the clock line due to ground bouncing or clock ringing due to over/undershoots would cause one or more extra bits to be written to the front end receive fifo of the slave. This extra bit, if no measures are taken by the software, would cause all the subsequent received frames to be corrupted (shifted).

The spike detection mechanism operates up to 50MBaud and checks if a bouncing spike has occurred on the clock line near to the correct receiving clock edges.

If a spike is detected, the software should use **GLOBALCON**.RESETS bitfield, and should clear the corresponding status flags. Alternatively a complete module reset may be performed via registers **KRST0**/**KRST1**.

The spike detection is based on two mechanisms:

- detection of two close writes to the front end receive FIFO by detecting filling level of two
- using an inverted watchdog set to ca. 75% to 80% SCLKI clock period, which is the optimal range.

The inverted watchdog is a timer which starts to count triggered on the write event to the front-end receive fifo of the slave.During the time the watchdog counts, no edge is allowed to occur. An edge in the forbidden time raises a spike error event.

If a spike occures in the last 25% of the SCLKI period, when the watchdog has stopped counting, the watchdog will be triggered to count by the spike. The next regular edge will violate the forbidden time and raise a spike event.

The watchdog counts **ECON**.B + **ECON**.C time quantas. The following constraint applies: 50% *T_{SCLKI}<**ECON**.B + **ECON**.C <100%*T_{SCLKI}.

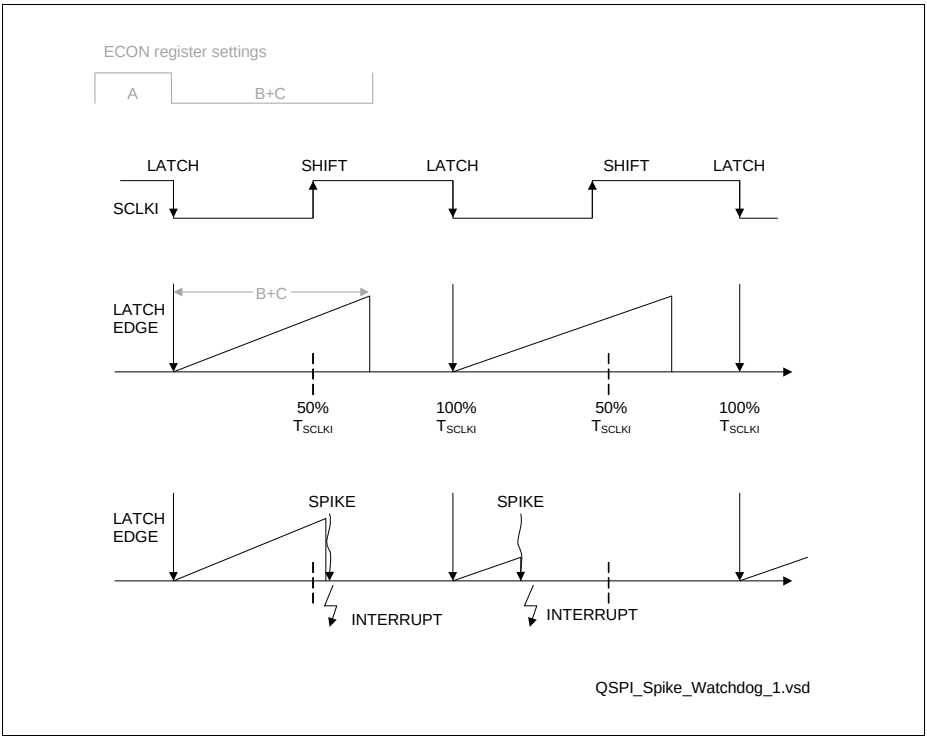


Figure 20-48 Spike Watchdog

20.8.2.3 Shift Clock Monitor Flags

The spike detection mechanism sets the flag **STATUS1**.SPD if it has detected a spike. The baud rate error detection mechanism sets the flag **STATUS1**.BRD if it has detected excessive baud rate deviation.

Writing 1 to SPD and BRD sets the bit and causes an interrupt if enabled. Writing 0 has no effect.

Both signals are ORed. This combined signal is shown in **STATUS**.ERRORFLAGS and raises an interrupt if enabled in **GLOBALCON1**.ERRORENS. Automatic clear of **GLOBALCON**.EN follows, if enabled by **GLOBALCON**.AREN.

If the **STATUS**.ERRORFLAGS[2] bit is cleared via **FLAGSCLEAR**.ERRORCLEARS[2], the both flags SPD and BRD are automatically cleared.

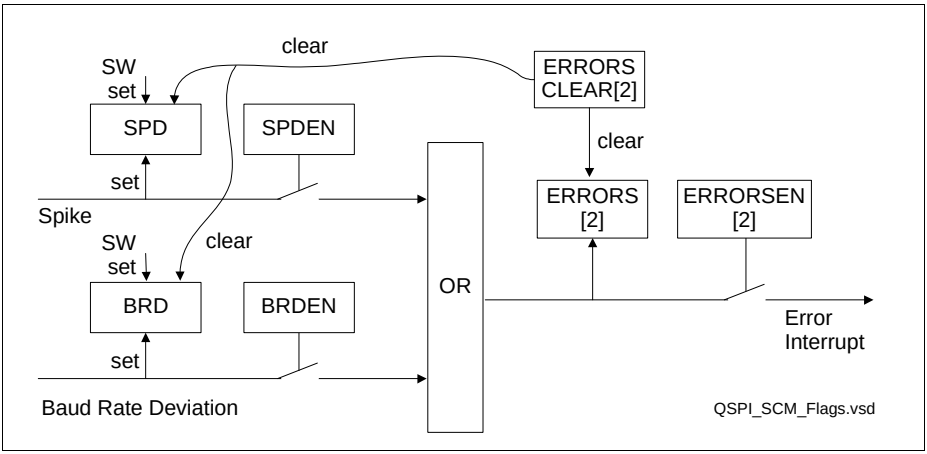


Figure 20-49 Shift Clock Monitor Flags

20.8.3 Parity

Parity settings for transmit and receive are defined by the corresponding bit fields in the **BACON** register.

The parity bit is concatenated to the data bits at transmission time. That means that a frame containing 16 bit data is 17 bits long (data + 1 parity). The transmitted parity bit can be read from the **STATUS**.TPV bit.

The parity bit is removed from the received bits at receive time automatically. The received parity bit can be read from the **STATUS**.RPV bit.

Attention: Parity is available only if the bit field BACON.BYTE = 0, that is, only for data of up to 32 bits length, that is, only in short mode.

Attention: The parity bit is concatenated to the data at the LSB location. If the bit field BACON.MSB = 0 it is shifted out first, else it is shifted out last.

20.9 Kernel Registers

This section describes the kernel registers of the QSPI module. All QSPI kernel register names described in this section will be referenced in other parts of the TC21x/TC22x/TC23x User's Manual by the module name prefix "QSPI0_" for the QSPI0 interface, "QSPI1_" for the QSPI1 interface and so on.

All registers in the QSPI address spaces are reset with the application reset (definition see SCU section "Reset Operation").

QSPI Kernel Register Overview

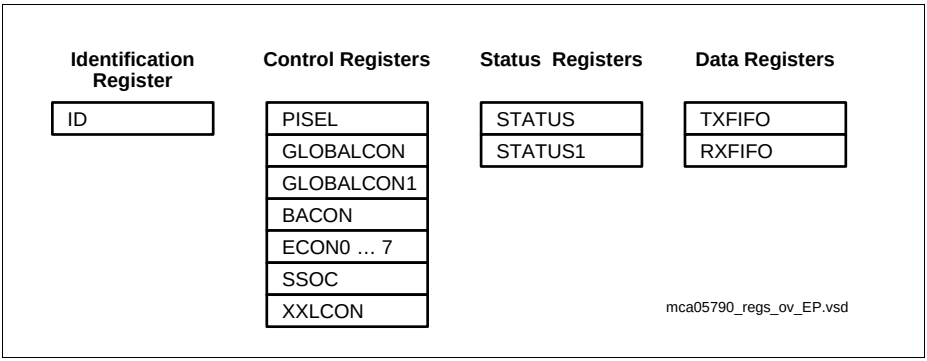


Figure 20-50 QSPI Kernel Registers

The following tables give the overview of the QSPI base addresses and registers.

Table 20-1 Registers Address Space

Module	Base Address	End Address	Note
QSPI0	$F0001C00_H$	$F0001CFF_H$	–
QSPI1	$F0001D00_H$	$F0001DFF_H$	–
QSPI2	$F0001E00_H$	$F0001EFF_H$	–
QSPI3	$F0001F00_H$	$F0001FFF_H$	–

Table 20-2 Registers Overview

Register Short Name	Register Long Name	Offset [1] Address	Write [2] Access	Reset Value (hex)
PISEL	Port Input Select Register	04_H	SV, P	0000 0000
ID	Module Identification Register	08_H	BE	00C0 C0XX

Table 20-2 Registers Overview (cont'd)

Register Short Name	Register Long Name	Offset [1] Address	Write [2] Access	Reset Value (hex)
reserved	reserved	$0C_H$	BE	
GLOBALCON	Global Configuration Register	10_H	SV, P	000F 30FF
GLOBALCON1	Global Configuration Register 1	14_H	SV, P	0005 0000
BACON	Basic Configuration Register	18_H	BE	0F87 1C71
reserved	reserved	$1C_H$	BE	
ECONz (z = 0 - 7)	Configuration Extension z Reg.	20_H...$3C_H$	SV, P	0000 1450
STATUS	Status Register	40_H	U, SV, P	0000 0000
STATUS1	Status Register 1	44_H	U, SV, P	0000 0000
SSOC	Slave Select Output Control R.	48_H	SV, P	0000 0000
reserved	reserved	$4C_H$	BE	
reserved	reserved	50_H	BE	
FLAGSCLEAR	Flags Clear Register	54_H	U, SV, P	0000 0000
XXLCON	Extra Large Data Config. Reg.	58_H	U, SV, P	0000 0000
MIXENTRY	MIX_ENTRY Register	$5C_H$	U, SV, P	0000 0000
BACONENTRY	BACON_ENTRY Register	60_H	U, SV, P	0000 0000
DATAENTRYx	Data Entry Addresses, x=0 to 7	64_H...80_H	U, SV, P	0000 0000
reserved	reserved	84_H...$8C_H$	BE	
RXEXIT	RX_EXIT Register	90_H	BE	0000 0000
RXEXITD	RX_EXIT Debug Register	94_H	BE	0000 0000
reserved	reserved	98_H...$9C_H$	BE	
CAPCON	Capture Control Register	$A0_H$	U,SV,P	0000 0000
reserved	reserved	$A4_H$...$E4_H$	BE	
BPI Registers				
CLC	Clock Control Register	00_H	SV, E, P	0000 0003
OCS	OCDS Control and Status Reg.	$E8_H$	SV, P	0000 0000
KRSTCLR	Reset Status Clear Register	EC_H	SV, E, P	0000 0000
KRST1	Reset Control Register 1	$F0_H$	SV, E, P	0000 0000
KRST0	Reset Control Register 0	$F4_H$	SV, E, P	0000 0000
ACCEN1	Access Enable Register 1	$F8_H$	SV, SE	0000 0000
ACCEN0	Access Enable Register 0	FC_H	SV, SE	FFFF FFFF

1) The absolute register address is Module Base Address (**Table 20-1**) + Offset Address (shown in this column)
2) All registers have read access mode U, SV. Read access to reserved addresses delivers bus error (BE). All registers belong to application reset except OCS, which belongs to debug reset.

List of Access Protection Abbreviations

U - User Mode

SV - Supervisor Mode

BE - Bus Error

nBE - no Bus Error

P - Access Protection, as defined by the ACCEN Register

E - ENDINIT

SE - SafetyENDINIT

20.9.1 Kernel Registers

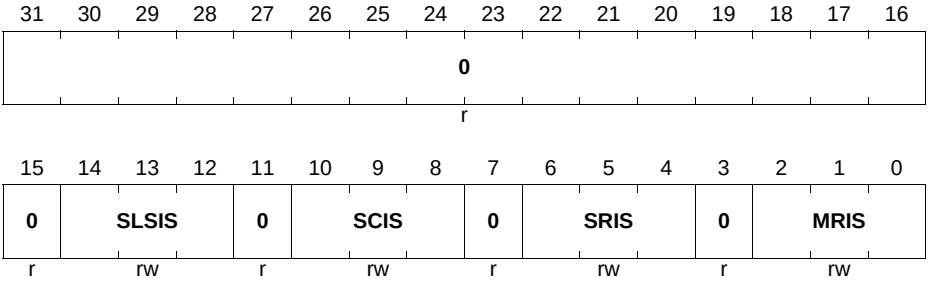
PISEL

The PISEL register controls the input signal selection of the SSC module.

(>> **Table 20-2** register overview)

PISEL

Port Input Select Register **(04_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
MRIS	[2:0]	rw	**Master Mode Receive Input Select** MRIS selects one out of eight MRST receive input lines (0 to 7 correspondingly), used in Master Mode. The signal suffixes “A” to “H” correspond to the bit field values of 0 to 7: MRSTxA ->0, MRSTxB->1 ...
SRIS	[6:4]	rw	**Slave Mode Receive Input Select** SRIS selects one out of eight MTSR receive input lines (0 to 7 correspondingly), used in Slave Mode. The signal sufixes “A” to “H” correspond to the bit field values of 0 to 7: MTSRxA ->0, MTSRxB->1 ...
SCIS	[10:8]	rw	**Slave Mode Clock Input Select** SCIS selects one out of eigjt module kernel SCLK input lines (0 to 7 correspondingly) that is used as clock input line in slave mode. The signal sufixes “A” to “H” correspond to the bit field values of 0 to 7: SCLKIxA ->0, SCLKIxB->1 ...

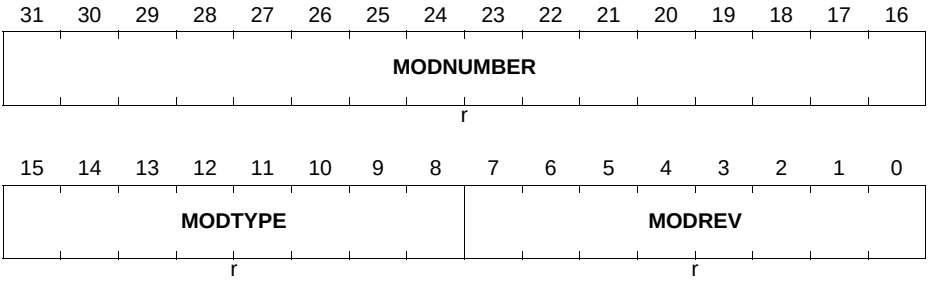
Field	Bits	Type	Description
SLSIS	[14:12]	rw	**Slave Mode Slave Select Input Selection** 000_B Slave select input lines are deselected; QSPI is operating without slave select input functionality. 001_B $\overline{SLSI}$ input line A is selected for operation 010_B $\overline{SLSI}$ input line B is selected for operation 011_B $\overline{SLSI}$ input line C is selected for operation 100_B $\overline{SLSI}$ input line D is selected for operation 101_B $\overline{SLSI}$ input line E is selected for operation 110_B $\overline{SLSI}$ input line F is selected for operation 111_B $\overline{SLSI}$ input line G is selected for operation The SLSIS must be programmed properly before the slave mode is set with **GLOBALCON**.MODE and the module is set to RUN mode.
0	[31:15], 11, 7, 3	r	**Reserved** Read as 0; should be written with 0.

ID

The Module Identification Register ID contains read-only information about the module version.

(>> **Table 20-2** register overview)

ID
Module Identification Register (08$_H$) Reset Value: 00C0 C0XX$_H$



Field	Bits	Type	Description
MODREV	[7:0]	r	**Module Revision Number** MODREV defines the module revision number. The value of a module revision starts with 01$_H$ (first revision).
MODTYPE	[15:8]	r	**Module Type** This bit field is C0$_H$. It defines a 32-bit module.
MOD NUMBER	[31:16]	r	**Module Number Value** This bit field together with MODTYPE uniquely identifies a module.

GLOBALCON

GLOBALCON contains configuration parameters which affect all channels.

(>> **Table 20-2** register overview)

GLOBALCON
Global Configuration Register (10_H) Reset Value: 000F 30FF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESETS				ARE N	MS		EN	0	STIP	SRF	STROBE				
w				rw	rw		rwh	r	rw	rw	rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEL 0	LB	EXPECT				SI	0	TQ							
rw	rw	rw				rw	r	rw							

Field	Bits	Type	Description
TQ	[7:0]	rw	**Global Time Quantum Length** Common n-divider scaling the baud rates of all channels in direction of higher or lower baud rates. Must not be changed during a running transaction. 0_D division by 1 1_D division by 2 $\ldots_D$ ... 255_D division by 256
SI	9	rw	**Status Injection** Selects if the status register content injection into the RxFIFO is enabled or disabled. The status injections, if enabled, is performed after each data block, depending on the **BACON**.DL and BYTE setting. 0_B disabled 1_B enabled
EXPECT	[13:10]	rw	**Time-Out Value for the Expect Phase** expressed in T_{QSPI} units 0_D 64 (2^6) 1_D 128(2^7) $\ldots_D$ ... 15_D 2097152 (2^{21} = 2 Mega)

Field	Bits	Type	Description
LB	14	rw	**Loop-Back Control** Selects if the transmit output is internally connected to the receive input for test purposes. 0_B Loop-Back inactive 1_B Loop-Back active For detailed description, see the **Loop-Back Mode** section.
DEL0	15	rw	**Delayed Mode for SLSO0** Switches the delayed mode for external multiplexer enabling on and off 0_B Delayed mode off 1_B Delayed mode on
STROBE	[20:16]	rw	**Strobe Delay for SLSO0 in Delayed Mode** Defines the length of the SLSO0 delay in T_Q time units as defined for channel z (T_Q units) selected by the current **BACON**.CS, if **GLOBALCON**.DEL0 = 1. 00000_B 1 00001_B 2 $\ldots_B$ ... 11111_B 32
SRF	21	rw	**Stop on RxFIFO Full** If this bit is set, the data fetching out of the TxFIFO by the shift register stops when the RxFIFO is full, in order to prevent RxFIFO overflow. Master mode only. 0_B Feature disabled 1_B Feature enabled
STIP	22	rw	**Slave Transmit Idle State Polarity** This bit determines the logic level of the Slave Mode transmit signal MRST when the QSPI slave select input signals are inactive (PISEL.SLSIS $\neq 000_B$). 0_B MRST = 0 when QSPI is deselected in Slave Mode. 1_B MRST = 1 when QSPI is deselected in Slave Mode

Field	Bits	Type	Description
EN	24	rwh	**Enable Bit** Used to request transition between PAUSE and RUN mode per software. Cleared by hardware automatically at leaving the following states: disabled, suspend and sleep. In order to determine if the requested state has actually been reached, the **STATUS**.PHASE bit field should be polled. See also **Operation Modes**. 0_B PAUSE requested 1_B RUN requested
MS	[26:25]	rw	**Master Slave Mode** Selects if the module operates in master or slave mode. This bit field must be configured before the first write to the TXFIFO. 00_B Master Transmit and Receive 01_B Reserved 10_B Slave Transmit and Receive 11_B Slave Transmit and Receive
AREN	27	rw	**Automatic Reset Enable** Enables the reset of the **GLOBALCON**.EN on baud rate and spike error in slave mode. 0_B disabled 1_B enabled
RESETS	[31:28]	w	**Bits for resetting sub-modules per software** Write to this bit field triggers a reset operation. The reset operation is not a single cycle operation, but takes several clock cycles, not more than the least common multiple of CCUCON0.SPBDIV and CCUCON0.BAUD2DIV (the configured division ratios, not the content of the bit fields). 0000_B No reset triggered 0111_B State Machine, TXFIFO and RXFIFO reset, registers not reseted 1111_B Module reset **others**, reserved For resetting the whole module, use **KRST0** / **KRST1** reset mechanism.

Field	Bits	Type	Description
0	8, 23	r	**Reserved** Read as 0; should be written with 0.

Note: If the EN bit is cleared first, and then the partial state machine reset activated, then the state machine goes into PAUSE state.

GLOBALCON1

GLOBALCON1 contains bit fields for control of the extension of the slave select signals by using an external demultiplexer.

(>> **Table 20-2** register overview)

GLOBALCON1
Global Configuration Register 1 (14_H) Reset Value: $0005\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**RXFM**		**TXFM**		**PT2**			**PT1**			**RXFIFO INT**		**TXFIFO INT**	
r	r	rw		rw		rw			rw			rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USR EN	**0**	**0**	**PT2 EN**	**PT1 EN**	**RX EN**	**TX EN**	**ERRORENS**								
rw	r	r	rw	rw	rw	rw	rw								

Field	Bits	Type	Description
ERRORENS	[8:0]	rw	**Error Enable Bits** Bits for enabling interrupt on all available error types 000000000_B All errors disabled 000000001_B Parity Error (PAREEN) 000000010_B Unexpected Configuration Error 000000100_B Baud Rate Error (slave mode) BEN 000001000_B TXFIFO overflow (software error) 000010000_B TXFIFO underflow (slave mode) TEN 000100000_B RXFIFO overflow REN 001000000_B RXFIFO underflow (software error) 010000000_B EXPECT timeout 100000000_B SLSI misplaced inactivation enable
TXEN	9	rw	**Tx Interrupt Event Enable** Enables the Tx interrupt. 0_B Disabled 1_B Enabled
RXEN	10	rw	**RX Interrupt Event Enable** Enables the Rx interrupt. 0_B Disabled 1_B Enabled

Field	Bits	Type	Description
PT1EN	11	rw	**Interrupt on PT1 Event Enable** Enables the PT interrupt on an PT1 event, as selected by the PT1 bit field. 0_B Disabled 1_B Enabled
PT2EN	12	rw	**Interrupt on PT2 Event Enable** Enables the PT interrupt on an PT2 event, as selected by the PT2 bit field. 0_B Disabled 1_B Enabled
USREN	15	rw	**Interrupt on USR Event Enable** Enables the USR interrupt on an USR event, as selected by the PT1 bit field. 0_B Disabled 1_B Enabled
TXFIFOINT	[17:16]	rw	**Transmit FIFO Interrupt Threshold** If the TXFIFO filling level is equal or less than this threshold, than each move of data or configuration from the TXFIFO triggers a transmit interrupt. Reset value of the level is 01_B. 00_B 1 01_B 2 10_B 3 11_B 4
RXFIFOINT	[19:18]	rw	**Receive FIFO Interrupt Threshold** If the RXFIFO filling level is equal or greater than this threshold, than each move of data or status (if enabled) into the RXFIFO triggers a receive interrupt. Reset value of the level is 01_B. 00_B 0 01_B 1 10_B 2 11_B 3

Field	Bits	Type	Description
PT1	[22:20]	rw	**Phase Transition Event 1** Selects the first phase transition to trigger the PT interrupt. 000_B BUSY (end of WAIT phase) 001_B SCLKPC (serial clock polarity change) 010_B SOF (Start of Frame) 011_B TBE (Transmit Buffer Emptied) 100_B RBF (Receive Buffer Filled) 101_B EOF (End of Frame) 110_B DNA (Data not Available = Start of Expect) 111_B CONT (End of EXPECT phase)
PT2	[25:23]	rw	**Phase Transition Event 2** Selects the second phase transition to trigger the PT interrupt. In master mode, the following events are available: 000_B BUSY (end of WAIT phase) 001_B SCLKPC (serial clock polarity change) 010_B SOF (Start of Frame) 011_B TBE (Transmit Buffer Emptied) 100_B RBF (Receive Buffer Filled) 101_B EOF (End of Frame) 110_B DNA (Data not Available = Start of Expect) 111_B CONT (End of EXPECT phase) In slave mode, the SLSI deactivated event is the only option available for PT2. For this purpose, always use the setting 101 (EOF).
TXFM	[27:26]	rw	**TXFIFO Mode** Selects between the Single Move Mode and the Batch Move Mode. 00_B Combined Move Mode 01_B Single Move Mode 10_B Batch Move Mode 11_B reserved
RXFM	[29:28]	rw	**RXFIFO Mode** Selects between the Single Move Mode and the Batch Move Mode. 00_B Combined Move Mode 01_B Single Move Mode 10_B Batch Move Mode 11_B reserved

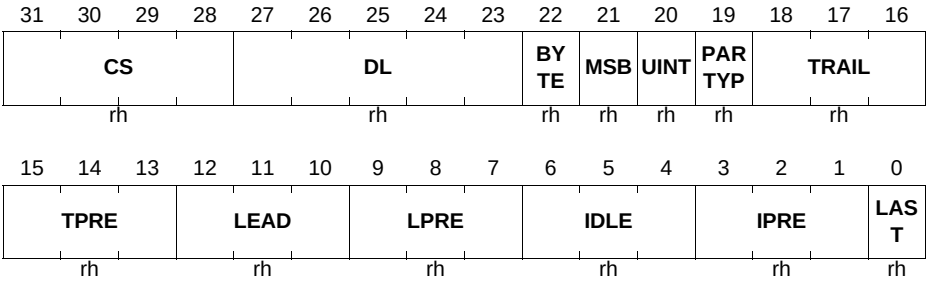
Field	Bits	Type	Description
0	30, 31, 13,14	r	**Reserved** Read as 0; should be written with 0.

BACON

Defines the basic configuration parameters for the current slave select. It can be read by software, or written through a write to the TXFIFO.

(>> **Table 20-2** register overview)

BACON
Basic Configuration Register (18_H) Reset Value: 0F87 1C71$_H$



Field	Bits	Type	Description
LAST	0	rh	**Last Word in a Frame** Defines if the following data word is last in the current frame or not 0_B Not Last 1_B Last
IPRE	[3:1]	rh	**Prescaler for the Idle Delay** Length in T_{BAUD2} units 000_B 1 001_B 4 010_B 16 …$_B$ ... 111_B 16384
IDLE	[6:4]	rh	**Idle Delay Length** Defines the length of both idle delays, IDLEA and IDLEB, in T_{BAUD2} units pre scaled with IPRE 0_D 1 units 1_D 2 unit …$_D$ ... 7_D 8 units

Field	Bits	Type	Description
LPRE	[9:7]	rh	**Prescaler for the Leading Delay** Length in T_{BAUD2} units 000_B 1 001_B 4 010_B 16 $\ldots_B$ ... 111_B 16384
LEAD	[12:10]	rh	**Leading Delay Length** Defines the length of the leading delay, in T_{BAUD2} units pre scaled with LPRE 0_D 1 units 1_D 2 unit $\ldots_D$ ... 7_D 8units
TPRE	[15:13]	rh	**Prescaler for the Trailing Delay** Length in T_{BAUD2} units 000_B 1 001_B 4 010_B 16 $\ldots_B$ ... 111_B 16384
TRAIL	[18:16]	rh	**Trailing Delay Length** Defines the length of the leading delay, in T_{BAUD2} units pre scaled with TPRE 0_D 1 units 1_D 2 unit $\ldots_D$ ... 7_D 8 units
PARTYP	19	rh	**Parity Type** Valid for both receive and transmit direction 0_B Even parity 1_B Odd parity

Field	Bits	Type	Description
UINT	20	rh	**User Interrupt at the PT1 Event in the Subsequent Frames** This bit is an enable signal for the PT1 event routed to the User Interrupt Service Request. The interrupt signals are generated until disabled with the next BACON. 0_B Disabled 1_B Enabled
MSB	21	rh	**Shift MSB or LSB First** This bit sets the shift direction of the shift register. If the MSB option is set, and the data is a block longer than 32 bits, the block must be fed into the TXFIFO in reverse direction, from the end of the block until its beginning. 0_B Shift LSB first 1_B Shift MSB first
BYTE	22	rh	**Byte** Defines if data length is expressed in bits or bytes 0_B DL defines the data length in bits 1_B DL defines the data length in bytes
DL	[27:23]	rh	**Data Length** Defines the data length in bits or bytes of one data block, depending on the setting of the bit field BYTE 0_D 2 bits if BYTE=0; XXL mode if BYTE=1 1_D 2 bits or bytes $\cdots_D$ ... 31_D 32 bits or bytes For the maximum baud rate of 50 MBaud, the minimum data length possible is four.
CS	[31:28]	rh	**Channel Select** Selects the channel to which the subsequent data entry belongs (channel = the SLSO signal to be activated and the corresponding **ECON** configuration extension) This bit field selects one slave in a range of 0 to 15, by driving one SLSO signal out of 16 available. In case of an external demux mode, this bit field appears on the lines SLS01 to SLSO4 as it is, additionally inverted or not, as defined in the **SSOC** register.

If an application does not use the SLSO signals of the QSPI module, but uses software channel selection, then the alternate function multiplexer in the ports must be configured so that the signal SLSO is not selected, but the coresponding bit Pn.OUT.

Alternatively, to emulate SLSO functionality, the software can disable (low) all the **SSOC**.OEN bits, and then toggle the **SSOC**.AOL bits.

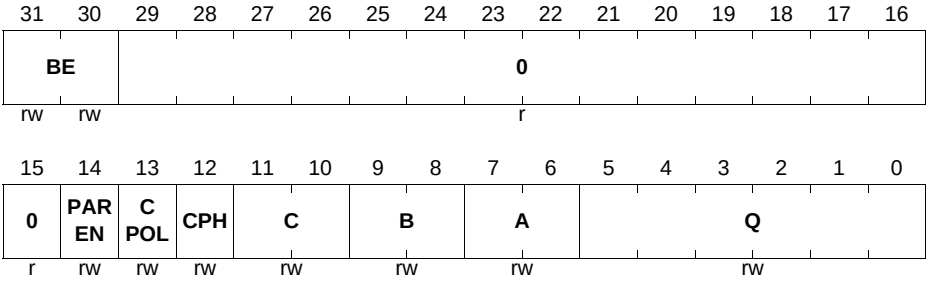
ECON

Configuration extensions for channels 0 to 15. Register x defines several timing characteristics for two channels: z and z+8.

(>> **Table 20-2** register overview)

(>> **Figure 20-12**, **Figure 20-13** timing diagrams)

ECONz (z = 0 - 7)
Configuration Extension z **(20_H + z*4_H)** **Reset Value: 0000 1450$_H$**



Field	Bits	Type	Description
Q	[5:0]	rw	**Time Quantum** Defines the time quantum length used by A, B, and C to define the baud rate and duty cycle by. This prescaler cascades the prescaler **GLOBALCON**.TQ. 000000_B 1 000001_B 2 $..._B$ ... 111111_B 64
A	[7:6]	rw	**Bit Segment 1** Length expressed in time quantums of **ECON**z.Q. 00_B 1 01_B 2 10_B 3 11_B 4

Field	Bits	Type	Description
B	[9:8]	rw	**Bit Segment 2** Length expressed in time quantums of ECONz.Q. 00_B 0 01_B 1 10_B 2 11_B 3
C	[11:10]	rw	**Bit Segment 3** Length expressed in time quantums of ECONz.Q. 00_B 0 (if B=0, than C is minimum 1 per hardware) 01_B 1 10_B 2 11_B 3
CPH	12	rw	**Clock Phase** Delay of one half SCLK clock cycle. 0_B Disabled 1_B Enabled
CPOL	13	rw	**Clock Polarity** Idle level of the shift clock signal at the SCLK pin 0_B Idle level low 1_B Idle level high
PAREN	14	rw	**Enable Parity Check** This bit field enables both the parity generation in transmit and parity check in receive direction. 0_B Disabled 1_B Enabled
BE	[31:30]	rw	**Permutate bytes to / from Big Endian** 00_B Disabled 01_B 16-bit big endian 10_B 32-bit big endian 11_B Disabled
0	[29:15]	r	**reserved**

STATUS

Status register contains bits flagging the current status of the module and its sub-modules.

Note: It is not recommended to set the STATUS register flags per software for purposes other than testing. In such cases, use write instructions only, not read-modify-write instructions or bit instructions which compile to read-modify-write instructions.

Note: When using the RXFIFO status injection feature, only bits 22 to 31 reflect the status at the moment of injection, that is for the latest frame. Due to pipeline effects, the other bits contain not-latest information.

(>> **Table 20-2** register overview)

STATUS
Status Register **(40_H)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PHASE				TPV	RPV	SLAVESEL				RXFIFO LEVEL			TXFIFO LEVEL		
rh				rh	rh	rh				rh			rh		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USRF	0	0	PT2F	PT1F	RXF	TXF	ERRORFLAGS								
rwh	r	r	rwh	rwh	rwh	rwh	rwh								

Field	Bits	Type	Description
ERROR FLAGS	[8:0]	rwh	**Sticky Flags Signalling Errors** Writing 1 sets the error Flag and triggers an error interrupt, if enabled. Writing 0 has no effect. 000000000_B No Error 000000001_B Parity Error 000000010_B Unexpected Configuration Error 000000100_B Baud Rate Error (slave mode) 000001000_B TXFIFO overflow (software error) 000010000_B TXFIFO underflow (slave mode) 000100000_B RXFIFO overflow 001000000_B RXFIFO underflow (software error) 010000000_B EXPECT time out error 100000000_B SLSI misplaced inactivation (slave mode)

Field	Bits	Type	Description
TXF	9	rwh	**Transmit Interrupt Request Flag** Flags an occurrence of a request to feed the TXFIFO, which is generated when an element is fetched from the FIFO, and the FIFO filling level is equal or less than the set threshold level. Writing 1 sets the flag and triggers an interrupt if **GLOBALCON1**.TXEN = 1. Writing 0 has no effect.
RXF	10	rwh	**Receive Interrupt Request Flag** Flags an occurrence of a request to empty the RXFIFO, which is generated when an element is written into the FIFO, and the FIFO filling level is equal or greater than the set threshold level. Writing 1 sets the flag and triggers an interrupt if **GLOBALCON1**.RXEN = 1. Writing 0 has no effect.
PT1F	11	rwh	**Phase Transition 1 Flag** Flags an occurrence of a PT1 event, as selected with the **GLOBALCON1**.PT1, and triggers an interrupt if **GLOBALCON1**.PT1EN = 1. Writing 1 sets the flag and triggers an error interrupt. Writing 0 has no effect.
PT2F	12	rwh	**Phase Transition 2 Flag** In master mode, flags an occurrence of a PT2 event, as selected with the **GLOBALCON1**.PT2, and triggers an interrupt if **GLOBALCON1**.PT2EN = 1. In slave mode, set by the SLSI deactivated event. Writing 1 sets the flag and triggers an error interrupt. Writing 0 has no effect.
USRF	15	rwh	**User Interrupt Request Flag** Flags an occurrence of an USR event. Writing 1 sets the flag and triggers an interrupt if **GLOBALCON1**.USREN = 1. Writing 0 has no effect.

Field	Bits	Type	Description
TXFIFO LEVEL	[18:16]	rh	**TXFIFO Filling Level** Shows how many entries in the TXFIFO are waiting for transmission 000_B 0 001_B 1 010_B 2 011_B 3 100_B 4 $..._B$ reserved
RXFIFO LEVEL	[21:19]	rh	**RXFIFO Filling Level** Shows how many entries in the RXFIFO are waiting for software to move them to RAM 000_B 0 001_B 1 010_B 2 011_B 3 100_B 4 $..._B$ reserved
SLAVESEL	[25:22]	rh	**Currently Active Slave Select Flag** Displays the currently active slave select.
RPV	26	rh	**Received Parity Value** Displays the last received parity bit, if parity was enabled. Else if the parity is disabled, reads 0.
TPV	27	rh	**Transmitted Parity Value** Displays the last transmitted parity bit, if parity was enabled. Else 0.

Field	Bits	Type	Description
PHASE	[31:28]	rh	**Flags the ongoing phase** Displays the current phase number. Relevant only in master mode. In slave mode this bit field indicates always 0. 0000_B Wait 0001_B Idle A 0010_B Idle B 0011_B Lead 0100_B Data 0101_B Trail 0110_B Expect 0111_B Lead Strobe 1000_B Trail Strobe Not 0 means busy.
0	13, 14	r	**Reserved** Read as 0; should be written with 0.

Note: Slave TXFIFO underflow error is activated if the transmission has been started by the master at the time when the slave FIFO was empty, or at the same moment updated. In the second case inconsistent data will be transmitted.

Note: Reading the TXFIFO filling level bit field returns a value which can be used to calculate how many writes can be performed by a CPU without causing an overflow (in case no DMA is programmed to access the TXFIFO in parallel). For example, if the TXFIFO level was one, maximum three (TXFIFO depth of four minus one) write accesses are possible. Due to the volatility of the bit field, the filling level can go down in some nanoseconds after the read.

Note: Reading the RXFIFO filling level bit field returns a value which shows directly how many reads can be performed by a CPU without causing an underflow. Due to the volatility of the bit field, the filling level can go up in some nanoseconds after the read.

Note: Reading the PHASE bit field shows a previous phase of the frame, and a PT1F and PT2F flags indicate previous phase transitions.
If the communication speed is not too high or the phases duration not very short compared to the software latency delays, which would normally be the case, these would be the latest completed phase and the latest phase transitions.
Nevertheless, in case of high baud rates, the duration of the phases must be taken

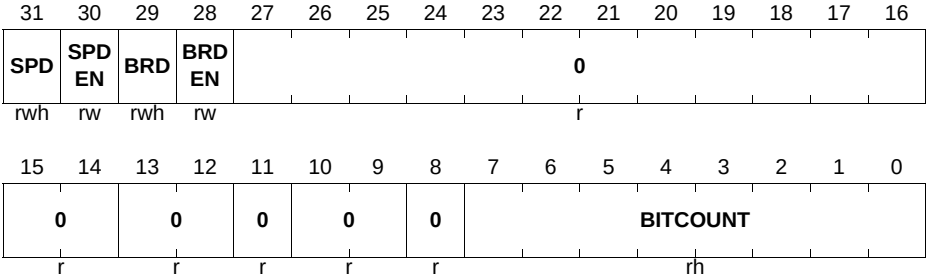
into consideration, but only if these bit fields are used in an application, and not only for debugging purposes.

STATUS1

(>> **Table 20-2** register overview)

STATUS1

Status Register 1 **(44_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
BITCOUNT	[7:0]	r	**Number of the bit shifted out** Supports up to 256 bits. BITCOUNT = 0 indicates two states: no transmission and transmission of the first bit. The differentiation can be made by reading the phase information. BITCOUNT = 1 indicates transmission of the second bit and so on until 256. After transmission of the last bit the counter goes to zero.
BRDEN	28	rw	**Baud Rate Deviation Enable** Enables the signal path. 0_B disabled 1_B enabled
BRD	29	rwh	**Baud Rate Deviation Flag** Shows if baud rate deviation has been detected. Write of 1 sets the bit and raises the event per software. Write of 0 has no effect. 0_B no event 1_B event detected
SPDEN	30	rw	**Spike Detection Enable** Enables the signal path. 0_B disabled 1_B enabled

Field	Bits	Type	Description
SPD	31	rwh	**Spike Detection Flag** Shows if spike has been detected. Write of 1 sets the bit and raises the event per software. Write of 0 has no effect. 0_B no event 1_B event detected
0	[27:8]	r	**Reserved** Read as 0; should be written with 0.

If the **STATUS**.ERRORFLAGS[2] bit is cleared via **FLAGSCLEAR**.ERRORCLEARS[2], the both flags SPD and BRD are automatically cleared.

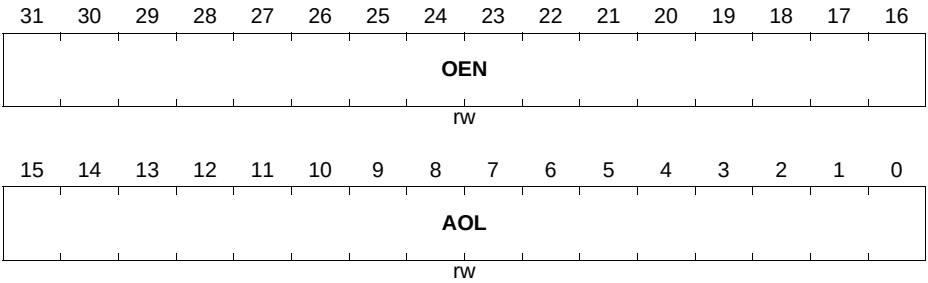
SSOC

SSOC controls the level of each slave select and enables/disables each one individually.

(>> **Table 20-2** register overview)

SSOC

Slave Select Output Control Register (48$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
OEN	[31:16]	rw	**Enable Bits for the SLSO Outputs** In disabled state the SLSO output drives the idle level as defined by the AOL bit field. “0” at certain position means that the corresponding SLSO is disabled. “1” means enabled.
AOL	[15:0]	rw	**Active Output Level for the SLSO Outputs** The idle level is the inverted one. “0” at certain position means active low level for the corresponding SLSO. “1” means active high.

FLAGSCLEAR

(>> **Table 20-2** register overview)

FLAGSCLEAR
Flags Clear Register **(54_H)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USRC	0	0	PT2C	PT1C	RXC	TXC	ERRORCLEARS								
w	r	r	w	w	w	w	w								

Field	Bits	Type	Description
ERROR CLEARS	[8:0]	w	**Write Only Bits for Clearing the Error Flags** Writing 1 clears the corresponding error flag in the ERORRFLAGS bit field. Reading returns 0. 000000000_B No clear 000000001_B Parity Error clear 000000010_B Unexpected Configuration Error clear 000000100_B Baud Rate Error clear 000001000_B TXFIFO overflow clear 000010000_B TXFIFO underflow clear 000100000_B RXFIFO overflow clear 001000000_B RXFIFO underflow clear 010000000_B EXPECT time out clear 100000000_B SLSI misplaced inactivation clear
TXC	9	w	**Transmit Event Flag Clear** Write of 1 clears the **STATUS**.TXF bit. Write of 0 has no effect. Read delivers 0. 0_B no action 1_B clear

Field	Bits	Type	Description
RXC	10	w	**Receive Event Flag Clear** Write of 1 clears the **STATUS**.RXF bit. Write of 0 has no effect. Read delivers 0. 0_B no action 1_B clear
PT1C	11	w	**PT1 Event Flag Clear** Write of 1 clears the **STATUS**.PT1F bit. Write of 0 has no effect. Read delivers 0. 0_B no action 1_B clear
PT2C	12	w	**PT2 Event Flag Clear** Write of 1 clears the **STATUS**.PT2F bit. Write of 0 has no effect. Read delivers 0. 0_B no action 1_B clear
USRC	15	w	**User Event Flag Clear** Write of 1 clears the **STATUS**.USRF bit. Write of 0 has no effect. Read delivers 0. 0_B no action 1_B clear
0	13, 14	rw	**Reserved** Read as 0; should be written with 0.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

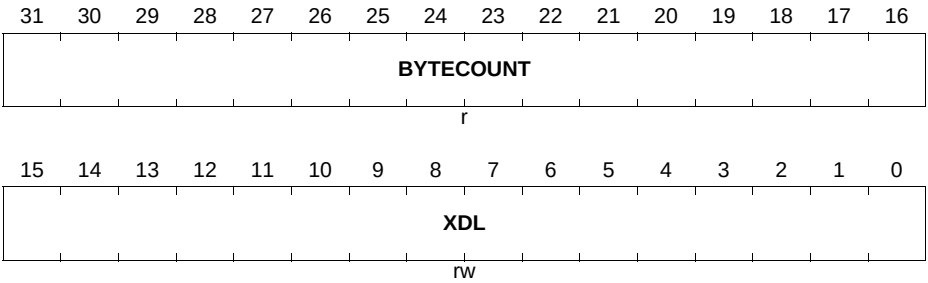
XXLCON

The XXLCON register provides counter for sending frames with very long blocks of data by extending the long data mode. It avoids a need for further BACON entries, like those needed in continuous mode.

Data length in this register overrides the **BACON**.DL setting in XXL mode, when **BACON**.BYTE=1 and **BACON**.DL=0. The data is sent to the slave as defined by **BACON**.CS bit field.

(>> **Table 20-2** register overview)

XXLCON
Extra Large Data Configuration Register(58_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
XDL	[15:0]	rw	**Extended Data Length** Defines the length of the data block in bytes in range of 2 to 65536. Overrides **BACON**.DL when **BACON**.BYTE=1 and and **BACON**.DL=0. 0_D 2 bytes 1_D 2 bytes $\ldots_D$ ... 65535_D 65536 bytes

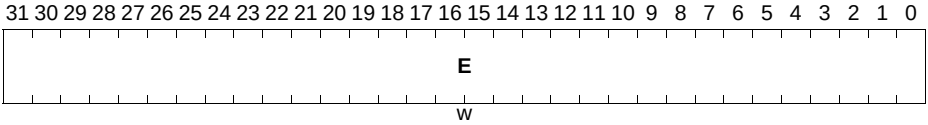
Field	Bits	Type	Description
BYTECOUNT	[31:16]	r	**Extended Data Length** In the XXL mode, shows the current state of the internal byte down counter (bytes remaining to be sent). In short and long modes, the value of this bit field is don't care. 0_D zero bytes 1_D two bytes $..._D$ ... 65535_D 65536 bytes

MIXENTRY

(>> **Table 20-2** register overview)

MIXENTRY

MIX_ENTRY Register **($5C_H$)** **Reset Value: 0000 0000$_H$**



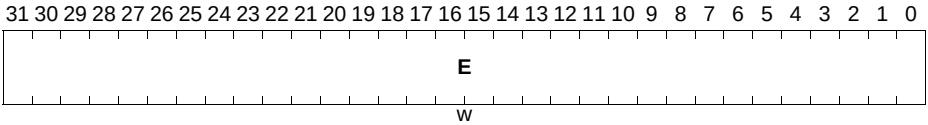
Field	Bits	Type	Description
E	[31:0]	w	**Entry Point to the TxFIFO**

BACONENTRY

(>> **Table 20-2** register overview)

BACONENTRY

BACON_ENTRY Register **(60_H)** **Reset Value: 0000 0000$_H$**



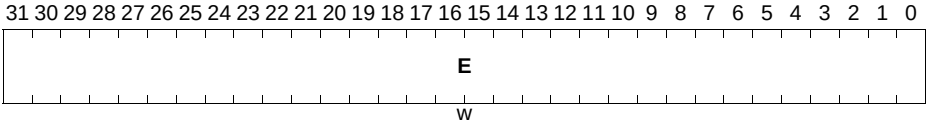
Field	Bits	Type	Description
E	[31:0]	w	**Entry Point to the TxFIFO**

DATAENTRY

(>> **Table 20-2** register overview)

DATAENTRYx (x=0-7)

DATA_ENTRY Register x **(64_H + x*4_H)** **Reset Value: 0000 0000_H**



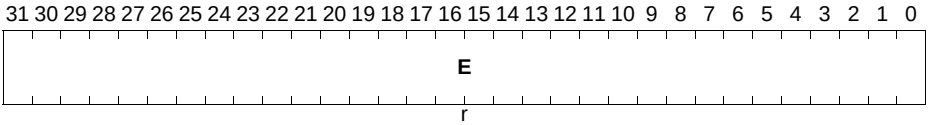
Field	Bits	Type	Description
E	[31:0]	w	**Entry Point to the TxFIFO**

RXEXIT

(>> **Table 20-2** register overview)

RXEXIT

RX_EXIT Register **(90_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
E	[31:0]	r	**Read Point from the RxFIFO**

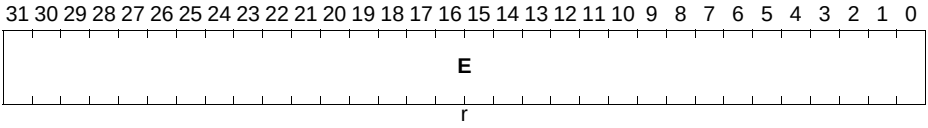
Note: The RXFIFO has a property that a read access from an empty RXFIFO generates an underflow interrupt, and delivers only “1” bits, which overrules the reset value. Therefore reading from a non initialized RXFIFO delivers all “1” and not all ”0”.

RXEXITD

The register **RXEXITD** provides a non-destructive address, showing the next available value in the RXFIFO.

(>> **Table 20-2** register overview)

RXEXITD
RX_EXIT Debug Register **(94_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
E	[31:0]	r	**Read Point from the RxFIFO**

*Note: This register provides a non-volatile access to the RXFIFO. It delivers the same value as **RXEXIT**, but without affecting read pointer and the filling level of the RXFIFO .*

20.10 Operation Modes

Generally, the QSPI module transmits a RAM queue of an arbitrary length, fed by a DMA module, and makes pauses between the queues. The QSPI module does not know anything about the queue length or the current position within the queue or the future schedule of the messages that may come.

If the QSPI module gets a request for switching off the clock or pausing the module, it disables all service request lines.

The QSPI module provides two options for switching off the clocks or pausing the module: hard suspend (as soon as possible regarding for example the length of interrupt request pulses), and soft suspend (transition to the PAUSE state with finishing the current frame). The requests can be triggered by hardware signals (disable, suspend, sleep) or by software write to **GLOBALCON**.EN bit.

Disable, Pause and Run

After being enabled by clearing the CLC.DISR bit, the QSPI enters the PAUSE state. In this state, the QSPI module can be initialized, the TXFIFO pre-filled, and then, by setting the **GLOBALCON**.EN bit, put into a RUN state.

The software requests PAUSE state of the running module by clearing the **GLOBALCON**.EN bit. The software can detect that the QSPI module has reached the PAUSE state by polling for the **GLOBALCON**.EN=0 and **STATUS**.PHASE=0 (indicating WAIT state, master mode only).

In PAUSE state, the user interface of the module is active. The registers and the FIFOs can be read and written. However, both the receive and transmit state machines are inactive, in master as well as in the slave mode. This allows for reconfiguration of the module without affecting the serial bus.

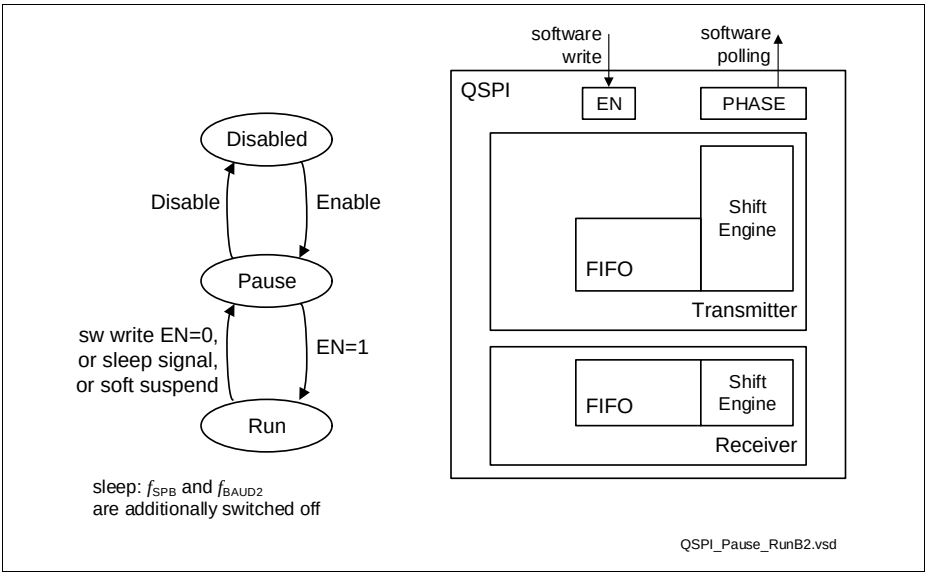


Figure 20-51 Pause and Run Overview

The concept of the PAUSE state is reused for handling the soft OCDS suspend and Sleep requests. In contrast to active Pause state and OCDS suspend, the sleep state has the special property that (due to the low power requirement) both FPI and QSPI clocks are switched off.

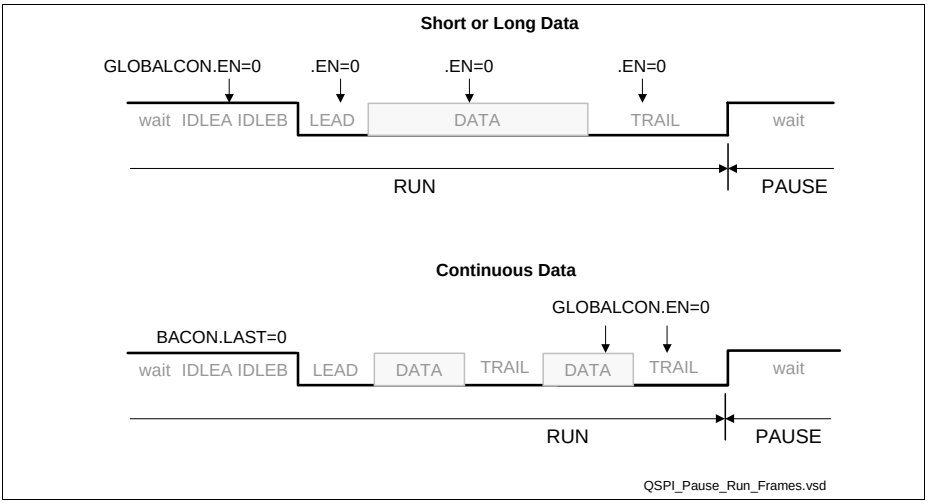


Figure 20-52 Entering the Pause State

If a running module receives a request to PAUSE, it waits for the end of the TRAIL phase.

For all modes (Short, Long and Continuous), the PAUSE state is indicated by **GLOBALCON**.EN=0 and **STATUS**.PHASE=0, master mode only.

20.10.1 OCDS Suspend

The OCDS hard suspend request, for debugging purposes, simply freezes the QSPI module in the current state. No signal is changed including the service request lines, which remain frozen in the current state.

The QSPI responds to an OCDS soft suspend request by entering the PAUSE state immediately after the subsequent end of a trail phase, and then sending acknowledge (see **Figure 20-53**). The behavior is the same as for entering the **Sleep Mode**.

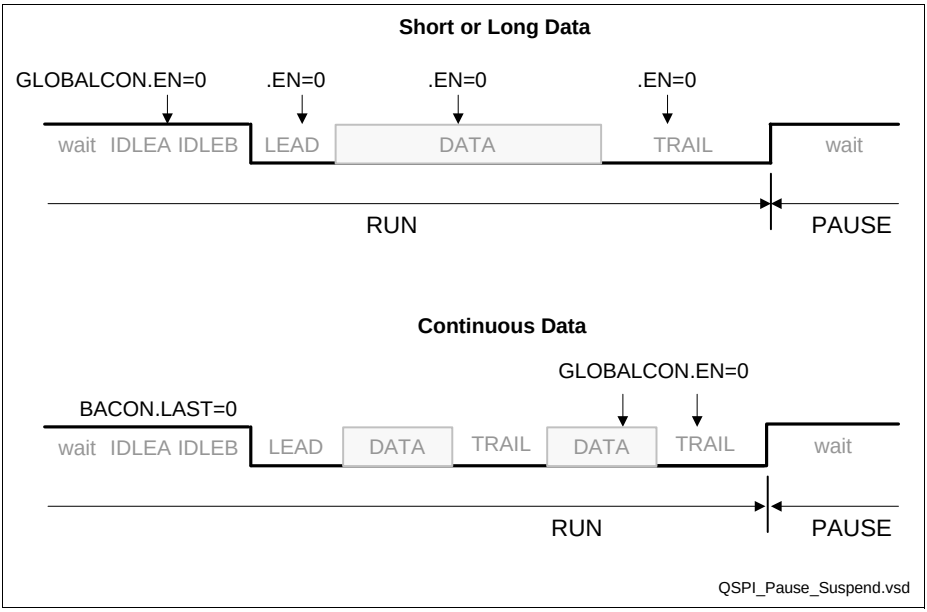


Figure 20-53 Entering the Sleep, Soft Suspend, and Disable State

Reading the FIFO entries by the Cerberus (detected by the FPI master tag) does not modify the FIFO content nor the read and write pointers.

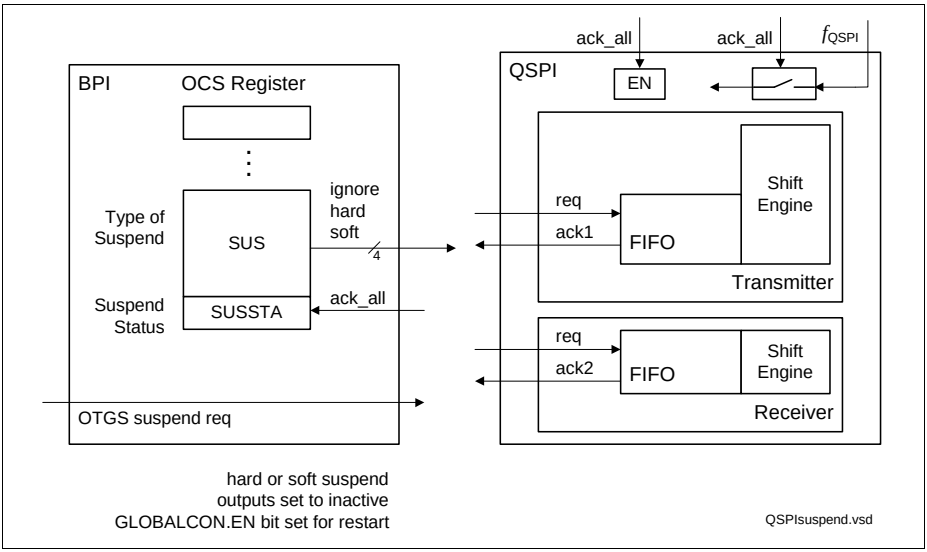


Figure 20-54 Suspend Overview

Note: In Hard Suspend Mode the QSPI kernel clock is switched off immediately. Reading and writing of registers is possible but will enable the kernel clock for a few cycles. Attention: register accesses with clocking in Hard Suspend Mode can have unintended side effects like signals becoming and staying active. This can affect also other modules, so a QSPI kernel reset might not be sufficient to bring the system into a defined state.

20.10.2 Sleep Mode

Sleep signal is a system wide hardware signal requesting from all modules to go to low power saving state as soon as possible. For the QSPI module, going to sleep mode can be disabled or enabled by setting or clearing the CLC.EDIS bit. The module enters the sleep as soon as possible and remains in this state as long as the hardware sleep signal is active. Upon deactivation of the sleep signal, the QSPI wakes in PAUSE state (**GLOBALCON**.EN bit is cleared by the QSPI module) and waits for the software to enable the **GLOBALCON**.EN bit.

If CLC.EDIS bit is cleared, the sleep mode is enabled and the module enters the sleep state:

in master mode

- after ending the TRAIL delay if a transmission is in progress
- or immediately if the module is in WAIT state

in slave mode

- after the bit counter has counted the full predefined number of bits as defined in GLOBALCON.DL and BYTE bit fields.
- immediately if the module is in wait state, no transmission ongoing and shift register empty and TXFIFO empty
- or immediately if the SLSI is inactive (the one selected by PISEL)

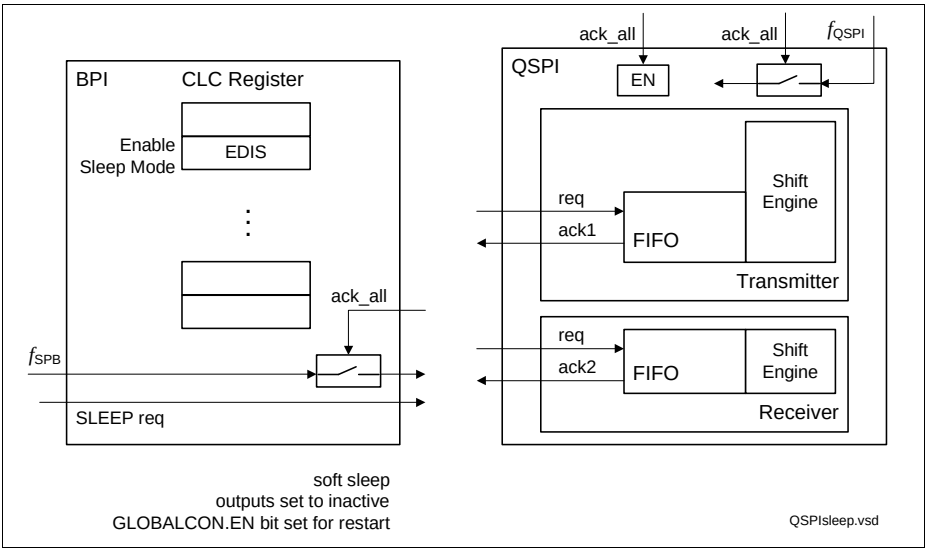


Figure 20-55 Sleep Overview

It is responsibility of the user software to have the sleep enabled in a safe time intervals, where no race conditions or pipeline effects between the QSPI module and the DMA can occur in case of sleep request.

Note: CLC disable request and sleep request block immediately kernel register write accesses. This includes also the TXFIFO access. Therefore if the module expects more data (like in long or continuous mode), it will enter the expect phase and remain there and never do an acknowledge and enter the required state. Periodic timeout interrupts will be generated. Therefore it is not recommended to request a sleep or disable during an ongoing long or continuous transfer.

20.10.3 Disabling the QSPI

The software can switch off the QSPI module by setting the CLC.DISR bit. It is responsibility of the user software to issue a disable request in a safe time interval, where no race conditions or pipeline effects between the QSPI module and the DMA can occur. This is no issue if the switch-off procedure is one-way, that is no continuation of operation without reinitialization/reset of the module is expected. **GLOBALCON**.EN bit is cleared.

The QSPI module itself behaves in the following way.

in master mode like entering the sleep mode:

- after ending the TRAIL delay if a transmission is in progress
- or immediately if the module is in WAIT state

in slave mode

- after the bit counter has counted the full predefined number of bits as defined in GLOBALCON.DL and BYTE bit fields, if a transmission is ongoing or
- immediately if the module is in wait state, no transmission ongoing and shift register empty and TXFIFO empty

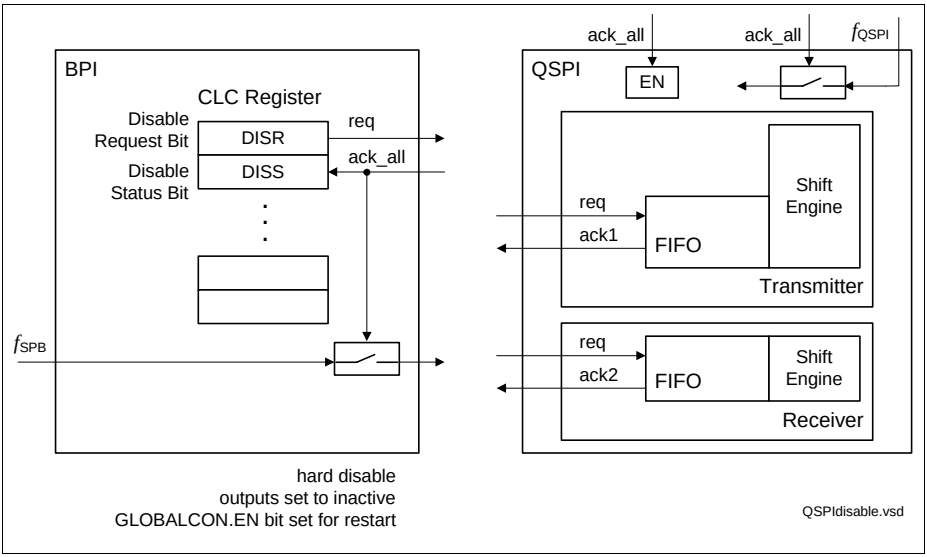


Figure 20-56 Disable Overview

20.11 Reset Behavior

There are two sources of reset:

- BPI (Bus Peripheral Interface)
- Reset bits in the kernel register - **GLOBALCON**.RESETS

BPI reset puts the whole QSPI module in reset state. All outputs get the reset value.

Software writing the **GLOBALCON**.RESETS bit field contains bits for resetting: TXFIFO, RXFIFO, shift state machine, register file. Setting all the bits in a single write resets the whole module.

20.12 QSPI Module Implementation

This section describes the interfaces of the QSPI module instances with the clock control, port connections, interrupt control, and address decoding. There are 4 module instances, QSPI0 to QSPI3.

20.12.1 Module Identification Registers

The reset values of the QSPIx_ID module identification registers are 00C0 C0XX$_H$.

20.12.2 Interfaces of the QSPI Modules

Figure 20-57 shows the implementation details and interconnections of the QSPI module.

Each of the QSPI modules is supplied with a separate clock control, interrupt control, and address decoding logic. Two interrupt outputs can be used to generate DMA requests. The QSPIx I/O lines are connected to the ports as described in the pinning and ports chapters.

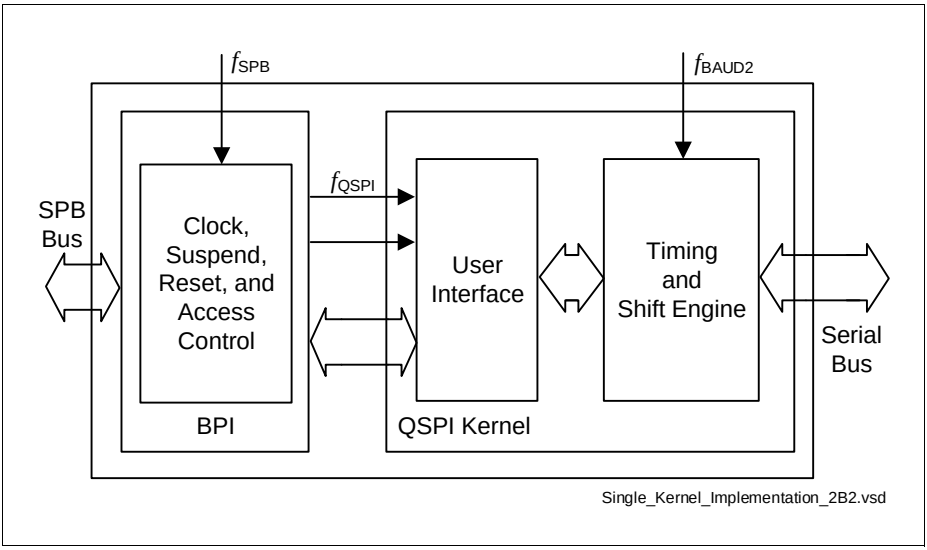


Figure 20-57 Module Implementation Overview

20.12.3 On-Chip Connections

This section describes the on-chip connections of the QSPIx module instances.

SCU Connections

Wake-up from sleep is done by the SCU.ERU triggering a software interrupt on a programmable edge on SLSI pin.

DMA Requests

There are no dedicated DMA request lines. The interrupt or service request signals are general purpose request signals that can be routed to each service provider.

Port/Pin Connections

The connections of the QSPI modules to the pins / ports is described in the chapters regarding the pinning and the ports.

The MTSR and MRST signals are also mapped to LVDS output and input pads. For details see the Ports chapter and the Pinning chapter.

Note: If LVDSH receiver pad is used, the software must take into account the corresponding control and configuration bits, located in the HSCT module and in the corresponding port registers.

Input signals not connected to ports/pins are connected to the following voltage levels:

- The unconnected SCLKI signals are connected to low level “0”
- The unconnected SLSI signals are connected to (inactive) high level “1”
- The unconnected MRST input signals are connected to low level “0”.

20.12.4 QSPI Related External Registers

Figure 20-58 summarizes the module-related external registers which are required for QSPI programming. These are:

- port registers and
- service request node registers,

described in the corresponding chapters.

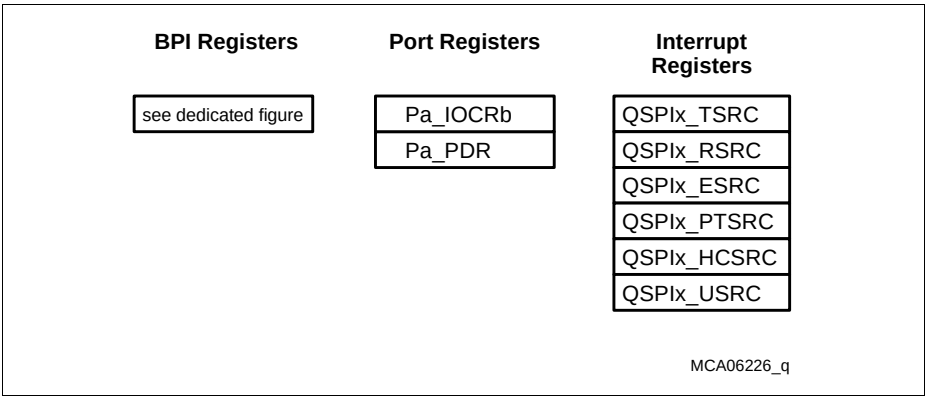


Figure 20-58 Implementation-specific Special Function Registers

20.12.4.1 BPI_FPI Module Registers

System Registers

Figure 20-59 shows all registers associated with the BPI_FPI module, configured for one kernel. The Offset Address in the following BPI_FPI register table and in the BPI_FPI register descriptions are proposals. In a standard 32 bit peripheral the CLC should be mapped to offset address 00h, module ID to 08h. The proposal for the new BPI_FPI register is to map them to the end of the peripheral address range.

BPI_FPI Registers Overview

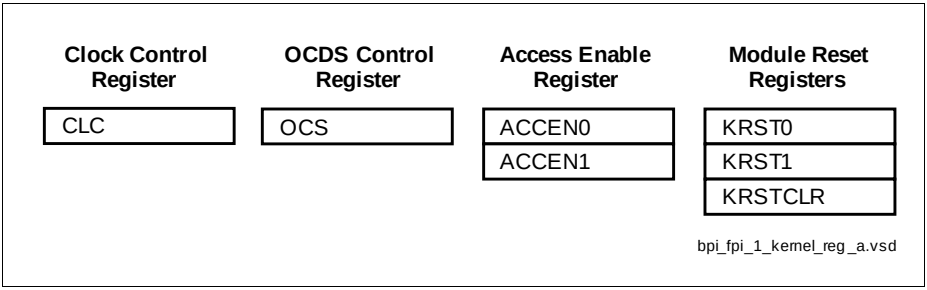


Figure 20-59 BPI_FPI Registers

The writes of the bus masters to the QSPI module is controlled by Access Protection registers ACCENx.

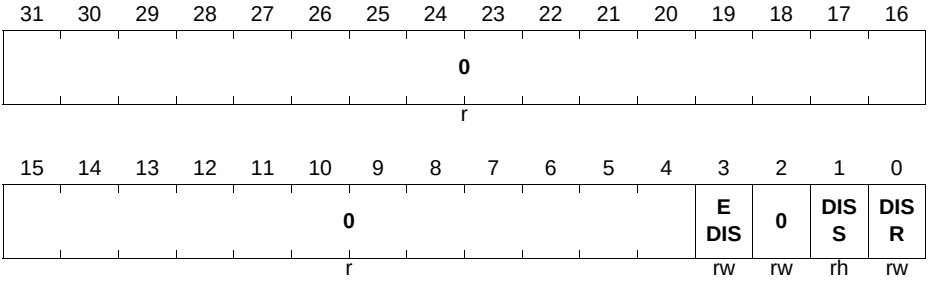
The QSPI implements two ACCENx registers, ACCEN0 and ACCEN1.

The ACCENx registers are protected by Safety Endinit mechanism.

Clock Control Register (CLC)

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI_FPI for the module. Where a module kernel is connected to the CLC clock control interface, CLC controls the $f_{<>}$ module clock signal, sleep mode and disable mode for the module.

CLC
Clock Control Register **(00_H)** **Reset Value: 0000 0003$_H$**



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module.
EDIS	3	rw	**Sleep Mode Enable Control** Used to enable the module's sleep mode. 0_B Enabled 1_B Disabled
0	[31:4]	r	**Reserved** Read as 0; should be written with 0.
0	2	rw	**Reserved** Should be written with 0.

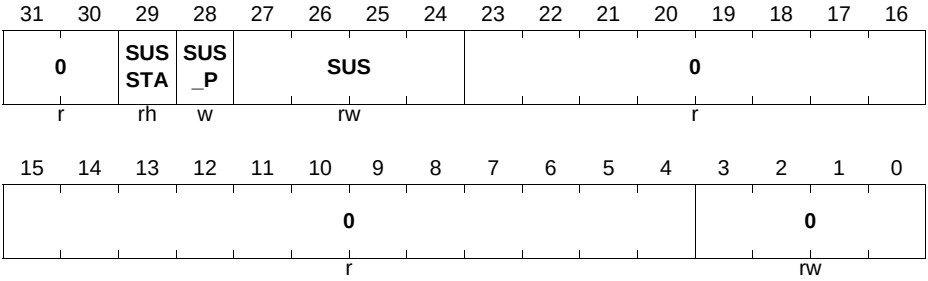
OCDS Control and Status Register (OCS)

The OCDS Control and Status (OCS) register is cleared by Debug Reset.

The register can only be written when the OCDS is enabled. If OCDS is being disabled, the OCS value will not change. When OCDS is disabled the OCS suspend control is ineffective. Write access is 32 bit wide only and requires Supervisor Mode

OCS

OCDS Control and Status (E8$_H$) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
SUS	[27:24]	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0_H Will not suspend 1_H Hard suspend. Clock is switched off immediately. 2_H Soft suspend **others**, reserved
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	[23:0], [31:30]	r	**Reserved** Read as 0; must be written with 0. The bits [4:0] are rw (readable and writable).

Access Enable Register (ACCEN0)

The Access Enable Register 0 controls write access[1)] for transactions with the on chip bus master TAG ID 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

1) The BPI_FPI Access Enable functionality controls only write transactions to the CLC, OCS, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000_B, EN1 -> TAG ID 000001_B, ... , EN31 -> TAG ID 011111_B.

ACCEN0
Access Enable Register 0 **(FC_H)** **Reset Value: FFFF FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN 31	**EN 30**	**EN 29**	**EN 28**	**EN 27**	**EN 26**	**EN 25**	**EN 24**	**EN 23**	**EN 22**	**EN 21**	**EN 20**	**EN 19**	**EN 18**	**EN 17**	**EN 16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN 15	**EN 14**	**EN 13**	**EN 12**	**EN 11**	**EN 10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

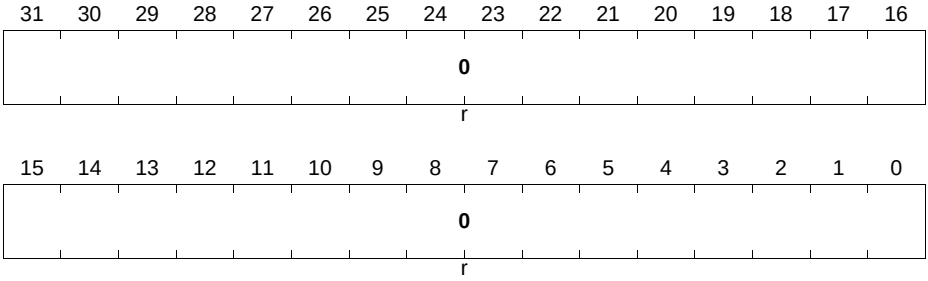
Field	Bits	Type	Description
ENn (n = 0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Access Enable Register (ACCEN1)

The Access Enable Register 1 controls write access[1)] for transactions with the on chip bus master TAG ID 100000_B to 111111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the AURIX devices.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001_B, ... , EN31 -> TAG ID 111111_B.

ACCEN1
Access Enable Register 1 **($F8_H$)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
0	[31:0]	r	**Reserved** Read as 0; should be written with 0.

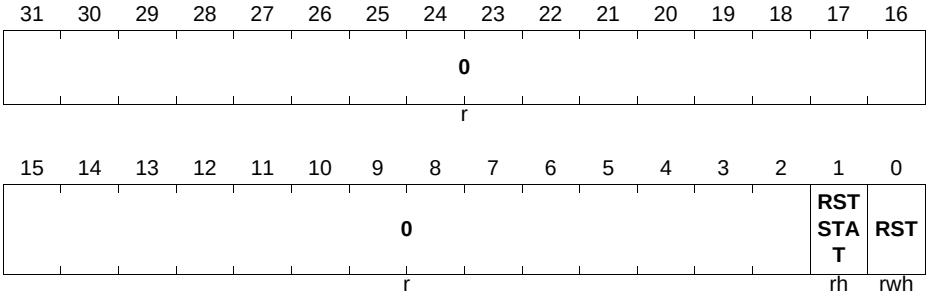
Kernel Reset Register 0 (KRST0)

The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers.The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the KRSTCLR.CLR register bit.

Note: During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

KRST0
Kernel Reset Register 0 **($F4_H$)** **Reset Value: 0000 0000$_H$**



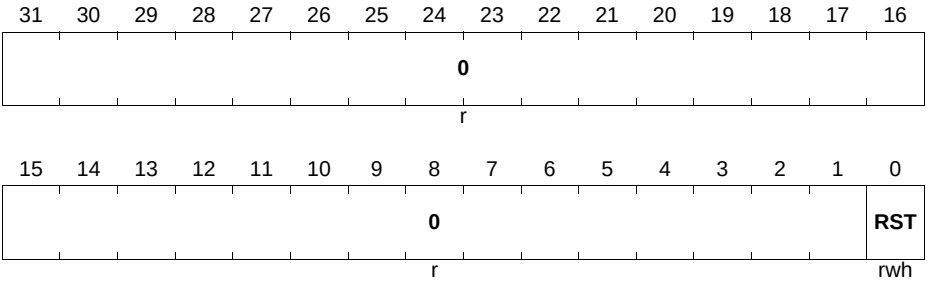
Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed.
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. 0_B No kernel reset was executed 1_B Kernel reset was executed This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register.
0	[31:2]	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1 (KRST1)

The Kernel Reset Register 1 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset registers. The RST

bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

KRST1
Kernel Reset Register 1 **($F0_H$)** **Reset Value: $0000\ 0000_H$**



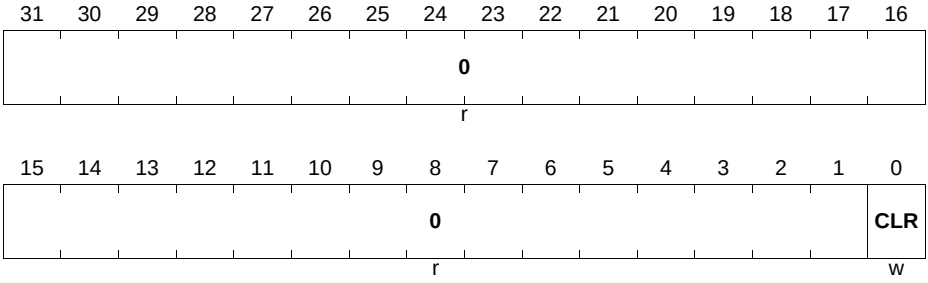
Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Status Clear Register (KRSTCLR)

The Kernel Reset Status Clear register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR

Kernel Reset Status Clear Register (EC$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT Read always as 0.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

20.12.4.2 Interrupt Control Registers

The interrupts of the QSPI module are controlled by six service request control registers, located in the IR (Interrupt Router) module:

- QSPIxTX Transmit Interrupt
- QSPIxRX Receive Interrupt
- QSPIxERR Error Interrupt
- QSPIxPT Phase Transition Interrupt
- QSPIxHC High Speed Capture Interrupt
- QSPIxU User Interrupt

20.13 High Speed Input Capture

The HSIC sub-module provides high precision measurement of relatively low speed input signals. It contains 15-bit timer can be configured to continuously measure:

- whole period (rising edge to rising edge),
- high time (rising edge to falling edge) or
- low time (falling edge to rising edge)
- whole period (falling edge to falling edge)

The resolution of the counter is f_{BAUD2}. The result is provided in a capture register and an interrupt is raised after each capture.

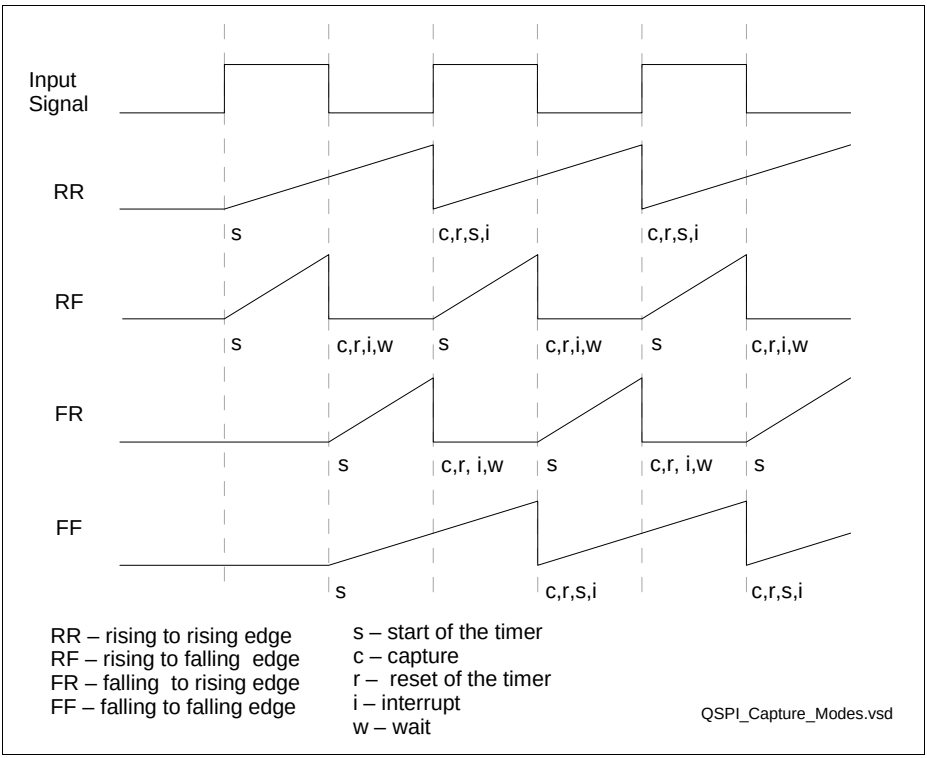


Figure 20-60 HSIC Modes of Operation

Each capture event causes the content of a 15-bit counter to be captured to the **CAPCON**.CAP bit field, see **Figure 20-61**.

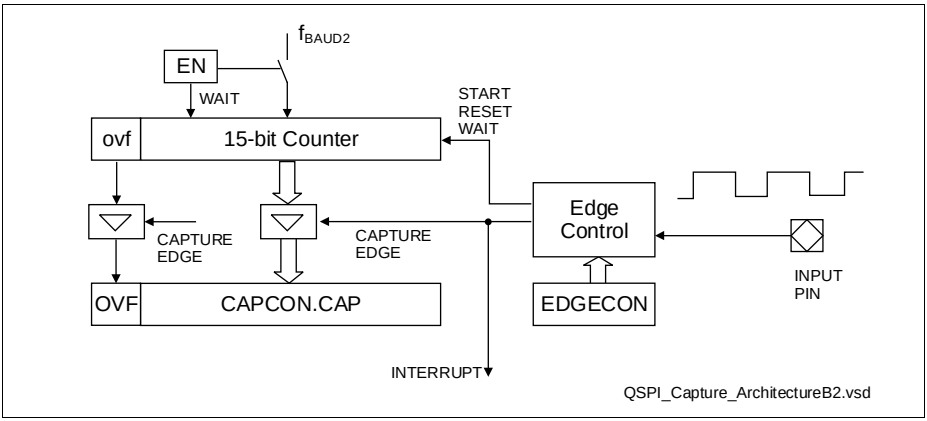


Figure 20-61 Capture Timer Architecture

The interrupt raised by the capture event is routed to the HC interrupt, see **Figure 20-62**. The flag bit, set, clear and select bits are located in the **CAPCON** register.

The first interrupt after enabling the HSIC block by setting **CAPCON**.EN may deliver random **CAPCON**.CAP value, depending on the level of the input signal at the moment of enabling. Therefore, the software should discard the first value captured immediately after enabling the HSIC block.

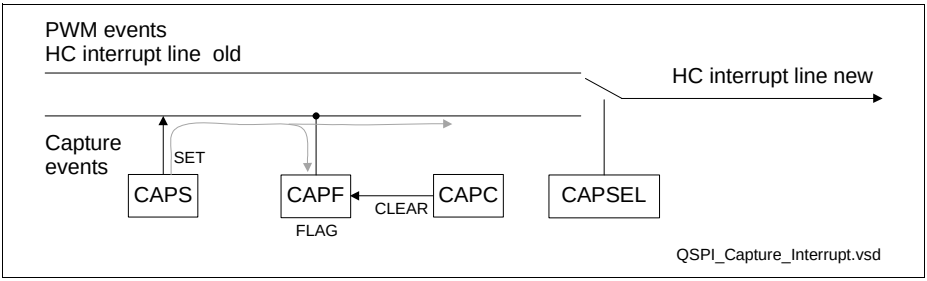


Figure 20-62 HSIC Interrupt Path

Disable Request

Clearing the disable request bit **CLC**.DISR switches the clock of the whole QSPI module including the HSIC block on.

Setting the bit **QSPI2_CAPCON**.EN enables the HSIC sub-module. The counter first enters wait state, waiting for the first start edge as defined in the **QSPI2_CAPCON**.EDGECON. Afterwards, it operates according to the **Figure 20-60**.

If **CLC**.DISR is cleared when **QSPI2_CAPCON**.EN = 1, the counter is cleared to zero, waits for the next start edge (as defined in **QSPI2_CAPCON**.EDGECON) to start counting, and continues operating according to the **Figure 20-60**.

Debug Suspend

Suspend of the QSPI module causes suspend of the HSIC; separate suspend is not possible. Nevertheless, the targeted use cases are either HSIC timer active, or QSPI communication, but not both.

Active and enabled hard debug suspend signal stops the HSIC block immediately (QSPI kernel disabled). After the suspend signal is deactivated, the counter continues counting. Writing to a register in hard suspend state causes the HSIC block to receive a couple of clock cycles, which causes the counter to increase its value.

Active and enabled soft debug suspend signal stops the HSIC block immediately (QSPI kernel disabled). After the suspend signal is deactivated, the counter is cleared to zero, waits for the next start edge (as defined in **QSPI2_CAPCON**.EDGECON) to start counting, and continues operating according to the **Figure 20-60**.

Sleep Mode

Active and enabled sleep signal stops the HSIC block. After the sleep signal is deactivated, the counter is cleared to zero, waits for the next start edge (as defined in **QSPI2_CAPCON**.EDGECON) to start counting, and continues operating according to the **Figure 20-60**.

Reset

Each reset of the QSPI module causes reset of the HSIC sub-module; separate reset is not possible.

20.13.1 HSIC Registers

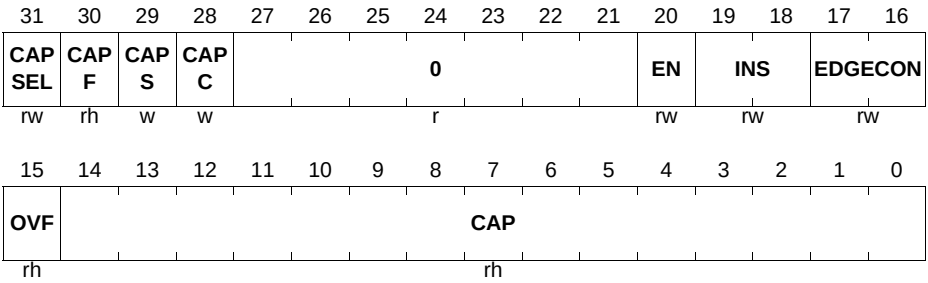
20.13.1.1 Register Description

CAPCON

CAPCON is the only control register for the HSIC sub-module. It contains the configuration, control bits and the captured value of the counter.

(>> **Table 20-2** register overview)

QSPI2_CAPCON
Capture Control Register **($A0_H$)** **Reset Value: 0000 0000$_H$**
QSPI3_CAPCON
Capture Control Register **($A0_H$)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CAP	[14:0]	rh	**Captured Value** Provides the latest capture value from the capture timer.
OVF	15	rh	**Overflow Flag** Signals if an overflow event of the capture timer occured in the latest measurement. Shows a value of 1 if one or more overflows of the capture timer occured during the latest measurement. Each capture event refreshes this bit with the new value.

Field	Bits	Type	Description
EDGECON	[17:16]	rw	**Edge Configuration** Configure the capture mode of the counter, definig which edge starts the counting, which edge captures its contents. 00_B RR mode (rising edge to rising edge) 01_B RF mode (rising edge to falling edge) 10_B FR mode (falling edge to rising edge) 11_B FF mode (falling edge to falling edge)
INS	[19:18]	rw	**Input Selection** Selects one of four input lines for the capture signal. 00_B INA selected 01_B INB selected 10_B INC selected 11_B IND selected
EN	20	rw	**Enable Bit of the Capture Timer** Enables the capture timer clock. Enabled timer waits for the next start edge as defined in EDGECON, before it starts counting. 0_B disabled 1_B enabled
CAPC	28	w	**Capture Flag Clear** Writing 1 to this bit clears the flag CAPF. Writing 0 has no effect. Returns 0 on read.
CAPS	29	w	**Capture Flag Set** Writing 1 to this bit sets the flag CAPF. Writing 0 has no effect. Returns 0 on read.
CAPF	30	rh	**Capture Flag** Indicates if a capture event has occured since the last clear event by the bit CAPC.
CAPSEL	31	rw	**Capture Interrupt Select Bit** Select if the capture events are routed to the HC interrupt line. 0_B not routed (capture interrupt disabled) 1_B routed (capture interrupt enabled)
0	[27:21]	r	**reserved**

20.13.2 HSIC Implementation

HSIC block is available in the QSPI2 and QSPI3.

20.13.2.1 Connection to the pins

For the connections to the pins, see the pinning chapter.

The unconnected input lines are tied to zero.

21 Controller Area Network Controller (MultiCAN+)

This chapter describes the MultiCAN+ controller of the TC21x/TC22x/TC23x. It contains the following sections:

- CAN basics (see **Page 21-2**)
- Overview of the CAN Module in the TC21x/TC22x/TC23x (see **Page 21-12**)
- CAN Flexible Data Rate in TC21x/TC22x/TC23x (see **Page 21-16**)
- Functional description of the MultiCAN+ Kernel (see **Page 21-18**)
- MultiCAN+ Kernel register description (see **Page 21-72**)
- TC21x/TC22x/TC23x implementation-specific details (port connections and control, interrupt control, address decoding, clock control, see **Page 21-152**).

Table 21-1 Fixed Module Constants

Constant	Description
n_objects	**Number of Message Objects available.**
n_interrupts	**Number of Interrupt Output Lines available.**
n_pendings **n_pendingregs**	**Number of Message Pending Bits available.** There are n_pendings/32 message pending registers.
n_lists	**Number of Lists available for allocation of Message Objects.**
n_nodes	**Number of CAN Nodes available** As each CAN node has its own list in addition to the list of un-allocated elements, the relation n_nodes < n_lists is true.

21.1 CAN Basics

CAN is an asynchronous serial bus system with one logical bus line. It has an open, linear bus structure with equal bus participants called nodes. A CAN bus consists of two or more nodes.

The bus logic corresponds to a “wired-AND” mechanism. Recessive bits (equivalent to the logic 1 level) are overwritten by dominant bits (logic 0 level). As long as no bus node is sending a dominant bit, the bus is in the recessive state. In this state, a dominant bit from any bus node generates a dominant bus state. The maximum CAN bus speed is, by definition, 1 Mbit/s. This speed limits the CAN bus to a length of up to 40 m. For bus lengths longer than 40 m, the bus speed must be reduced.

The binary data of a CAN frame is coded in NRZ code (Non-Return-to-Zero). To ensure re-synchronization of all bus nodes, bit stuffing is used. This means that during the transmission of a message, a maximum of five consecutive bits can have the same polarity. Whenever five consecutive bits of the same polarity have been transmitted, the transmitter will insert one additional bit (stuff bit) of the opposite polarity into the bit stream before transmitting further bits. The receiver also checks the number of bits with the same polarity and removes the stuff bits from the bit stream (= destuffing).

In CAN FD format frames, the CAN bit stuffing method is changed for the CRC Sequence. The stuff bits will be inserted at fixed positions.

21.1.1 Addressing and Bus Arbitration

In the CAN protocol, address information is defined in the identifier field of a message. The identifier indicates the contents of the message and its priority. The lower the binary value of the identifier, the higher is the priority of the message.

For bus arbitration, CSMA/CD with NDA (Carrier Sense Multiple Access/Collision Detection with Non-Destructive Arbitration) is used. If bus node A attempts to transmit a message across the network, it first checks that the bus is in the idle state (“Carrier Sense”) i.e. no node is currently transmitting. If this is the case (and no other node wishes to start a transmission at the same moment), node A becomes the bus master and sends its message. All other nodes switch to receive mode during the first transmitted bit (Start-Of-Frame bit). After correct reception of the message (acknowledged by each node), each bus node checks the message identifier and stores the message, if required. Otherwise, the message is discarded.

If two or more bus nodes start their transmission at the same time (“Multiple Access”), bus collision of the messages is avoided by bit-wise arbitration (“Collision Detection / Non-Destructive Arbitration” together with the “Wired-AND” mechanism, dominant bits override recessive bits). Each node that sends also reads back the bus level. When a recessive bit is sent but a dominant one is read back, bus arbitration is lost and the transmitting node switches to receive mode. This condition occurs for example when the message identifier of a competing node has a lower binary value and therefore sends a

message with a higher priority. In this way, the bus node with the highest priority message wins arbitration without losing time by having to repeat the message. Other nodes that lost arbitration will automatically try to repeat their transmission once the bus returns to idle state. Therefore, the same identifier can be sent in a Data Frame only by one node in the system. There must not be more than one node programmed to send Data Frames with the same identifier.

Standard message identifier has a length of 11 bits. CAN specification 2.0B extends the message identifier lengths to 29 bits, i.e. the extended identifier.

21.1.2 CAN Frame Formats

Four different data frame formats are supported which differ in the length of the Arbitration Field and Control Field:

- Classical CAN Base format: 11-bit long identifier, constant bit rate
- Classical CAN Extended format: 29-bit long identifier, constant bit rate
- CAN FD Base format: 11-bit long identifier, dual bit rate
- CAN FD Extended format: 29-bit long identifier, dual bit rate

In addition for Classical CAN remote frames exist, for 11-bit and 29bit identifiers.

21.1.3 CAN Frame Types

There are three types of CAN frames:

- Data Frames
- Remote Frames
- Error Frames

A Data Frame contains a Data Field of 0 to 8 bytes in length. A Remote Frame contains no Data Field and is typically generated as a request for data (e.g. from a sensor). Data and Remote Frames can use an 11-bit “Standard” identifier or a 29-bit “Extended” identifier. An Error Frame can be generated by any node that detects a CAN bus error.

21.1.3.1 Data Frames

There are four types of Data Frames defined (see **Figure 21-1**):

- Standard Data Frame Classical CAN Format
- Extended Data Frame Classical CAN Format
- Standard Data Frame CAN FD Format
- Extended Data Frame CAN FD Format

11-bit Data Frame (CAN Format)

A Data Frame begins with the Start-Of-Frame bit (SOF = dominant level) for hard synchronization of all nodes. The SOF is followed by the Arbitration Field consisting of 12 bits, the 11-bit identifier (reflecting the contents and priority of the message), and the

RTR (Remote Transmission Request for Classical CAN) bit. With RTR at dominant level, the frame is marked as Data Frame. With RTR at recessive level, the frame is defined as a Remote Frame.

The next field is the Control Field consisting of 6 bits. The first bit of this field is the IDE (Identifier Extension) bit and is at dominant level for the Standard Data Frame. The following bit is reserved and defined as a dominant bit. The remaining 4 bits of the Control Field are the Data Length Code (DLC) that specifies the number of bytes in the Data Field. The Data Field can be 0 to 8 bytes wide. The Cyclic Redundancy (CRC) Field that follows the data bytes is used to detect possible transmission errors. It consists of a 15-bit CRC sequence completed by a recessive CRC delimiter bit.

The final field is the Acknowledge Field. During the ACK Slot, the transmitting node sends out a recessive bit. Any node that has received an error free frame acknowledges the correct reception of the frame by sending back a dominant bit, regardless of whether or not the node is configured to accept that specific message. This behavior assigns the CAN protocol to the “in-bit-response” group of protocols. The recessive ACK delimiter bit, which must not be overwritten by a dominant bit, completes the Acknowledge Field.

Seven recessive End-of-Frame (EOF) bits finish the Data Frame. Between any two consecutive frames, the bus must remain in the recessive state for at least 3 bit times (called Inter Frame Space). If after the Inter Frame Space, no other nodes attempt to transmit the bus remains in idle state with a recessive level.

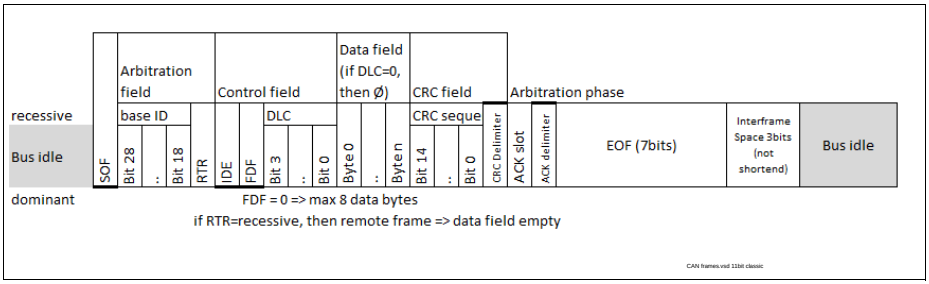


Figure 21-1 Classical 11bit ID CAN Data Frame

Extended Data Frame (Classical CAN Format)

In the Extended CAN Data Frame, the message identifier of the standard frame has been extended to 29-bit. A split of the extended identifier into two parts, an 11-bit least significant section (as in classical CAN frame) and an 18-bit most significant section, ensures that the Identifier Extension bit (IDE) can remain at the same bit position in both standard and extended frames.

In the Extended CAN Data Frame, the SOF bit is followed by the 32-bit Arbitration Field. The first 11 bits are the least significant bits of the 29-bit Identifier (“Base-ID”). These 11 bits are followed by the recessive Substitute Remote Request (SRR) bit. The SRR is

further followed by the recessive IDE bit, which indicates the frame to be an Extended CAN frame. If arbitration remains unresolved after transmission of the first 11 bits of the identifier, and if one of the nodes involved in arbitration is sending a classical CAN frame, then the CAN frame will win arbitration due to the assertion of its dominant IDE bit. Therefore, the SRR bit in an Extended CAN frame is recessive to allow the assertion of a dominant RTR bit by a node that is sending a CAN Remote Frame. The SRR and IDE bits are followed by the remaining 18 bits of the extended identifier and the RTR bit.

Control field and frame termination is identical to the Classical Data Frame.

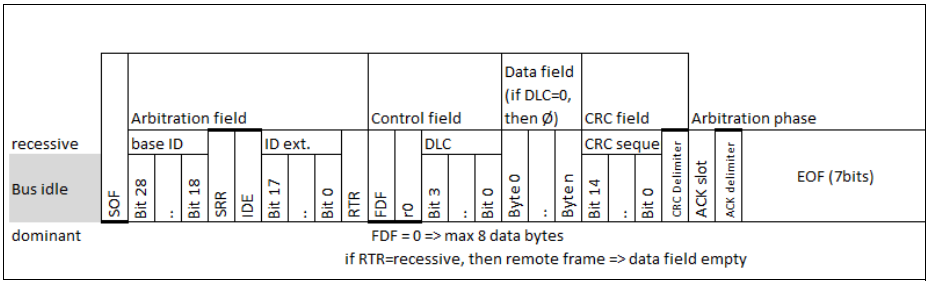


Figure 21-2 Classical 29 bit ID CAN Data Frame

Standard and Extended Data Frame (CAN FD Format)

Data frames for CAN FD with 11bit identifier are shown in **Figure 21-3**.

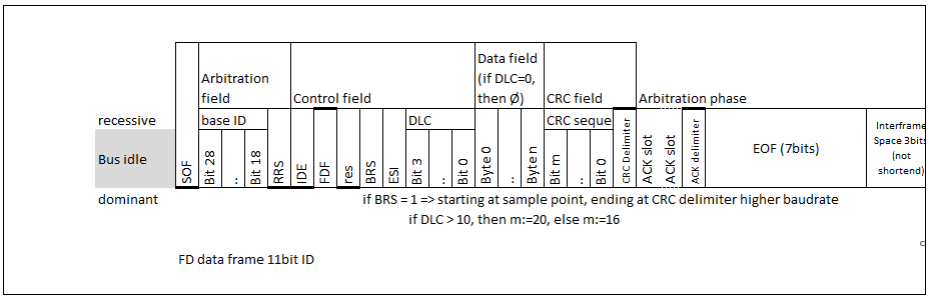


Figure 21-3 CAN FD 11bit ID Data Frames

Extended data frames for CAN FD are shown in **Figure 21-4**.

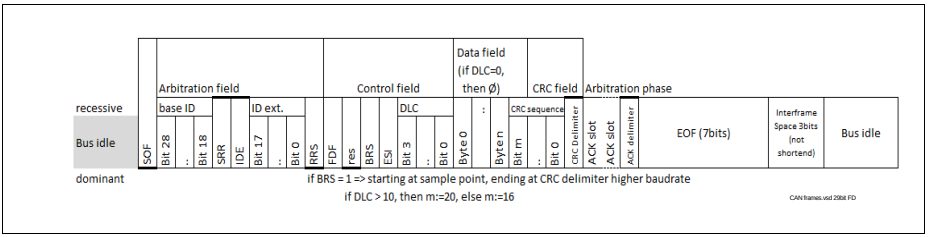


Figure 21-4 CAN FD 29bit ID Data Frames

The difference between Standard / Extended Data Frames in CAN and CAN FD format are highlighted below:

1. In the Arbitration Field:
   a) CAN Format Frames contain the RTR bit, where in CAN FD Format it is replaced with the dominant r1(reserved) bit.
   b) The reserved r0, r1 (bits) are sent dominant. Receivers accept dominant and recessive bits.
2. In the Control Field:
   a) CAN FD Format Frames consists of the additional (FDF) CAN FD format bit, (BRS) bit rate switch bit and (ESI) error state indicator bit.
   b) CAN FD format (FDF) bit comes after the IDE bit for frames with 11-bit identifier and it comes after the r1 (reserved) bit with 29-bit identifier. FDF is the new name for the previous r0 bit.
   c) Bit Rate Switch (BRS) bit switches the bit rate from standard bit rate of Arbitration Phase to preconfigured bit rate of the Data Phase when the bit is transmitted recessive. The bit rate is not switched when BRS bit is transmitted dominant.
   d) Error State Indicator (ESI) bit is transmitted dominant by error active nodes and recessive by error passive nodes.
   e) Data Length Code (DLC) bits indicates the number of bytes in the Data Field.
3. In the CRC Field:
   a) CRC sequence bits for CAN FD uses CRC_17 for data field up to sixteen bytes long and CRC_21 for data field longer than sixteen bytes. CRC calculation consists of the SOF, Arbitration Field, Control Field and (if present) Data field, supplemented with nCRC bits of ‘0’.
   Stuff-bits are included in CRC calculation. All CRC sequences (CRC_15, 17, 21) are calculated for all nodes, where the node that wins the arbitration send the CRC sequence selected by values of the FDF bit and DLC. Receivers only consider the selected CRC polynomial to check for CRC error
   b) CRC Delimiter for CAN FD consists of one or two recessive bits that has the function of switching the Data phase to Arbitration phase when the sample point reaches the first bit of the CRC Delimiter.
   Transmitter sends only one recessive bit as CRC delimiter, but accepts two

recessive bits before the edge from recessive to dominant of the Acknowledge slot. Receiver sends Acknowledge bit after the first CRC Delimiter.

c) Due to a weakness of the CRC in a certain situation, and additional checksum is needed. For example a software checksum inside the last data byte. This is a weakness in the standard and not of the module. Therefore with DIS2015 the CRC field of CAN FD will be changed.

4. In the ACK Field:
   a) For ACK slot, CAN FD nodes accept a two bit long dominant phase of overlapping ACK bits as a valid ACK, to compensate for phase shifts between Receivers.

21.1.3.2 Remote Frames

Normally, data transmission is performed on an autonomous basis with the data source node (e.g. a sensor) sending out a Data Frame. It is also possible, however, for a destination node (or nodes) to request the data from the source. For this purpose, the destination node sends a Remote Frame with an identifier that matches the identifier of the required Data Frame. The appropriate data source node will then send a Data Frame as a response to this remote request.

There are 2 differences between a Remote Frame and a Data Frame.

- The RTR bit is in the recessive state in a Remote Frame.
- There is no Data Field in a Remote Frame.

If a Data Frame and a Remote Frame with the same identifier are transmitted at the same time, the Data Frame wins arbitration due to the dominant RTR bit following the identifier. In this way, the node that transmitted the Remote Frame receives the requested data immediately.

Remote frames are only defined in Classical CAN format. Remote frames and the corresponding RTR bit do not exist for CAN FD format. Remote frame requests do not change the format of the preconfigured reply data frames (i.e FDF and BRS bits of preconfigured data frames are not changed on remote frame requests, reply data frames may be in CAN base/extended or CAN FD base/extended).

21.1.3.3 Error Frames

An Error Frame is generated by any node that detects a bus error. An Error Frame consists of two fields, an Error Flag field followed by an Error Delimiter field. The Error Delimiter Field consists of 8 recessive bits and allows the bus nodes to restart bus communications after an error. There are, however, two forms of Error Flag fields. The form of the Error Flag field depends on the error status of the node that detects the error.

When an error-active node detects a bus error, the node generates an Error Frame with an active-error flag. The error-active flag is composed of six consecutive dominant bits that actively violate the bit-stuffing rule. All other stations recognize a bit-stuffing error and generate Error Frames themselves. The resulting Error Flag field on the CAN bus

therefore consists of six to twelve consecutive dominant bits (generated by one or more nodes). The Error Delimiter field completes the Error Frame. After completion of the Error Frame, bus activity returns to normal and the interrupted node attempts to re-send the aborted message.

If an error-passive node detects a bus error, the node transmits an error-passive flag followed, again, by the Error Delimiter field. The error-passive flag consists of six consecutive recessive bits, and therefore the Error Frame (for an error-passive node) consists of 14 recessive bits (i.e. no dominant bits). Therefore, the transmission of an Error Frame by an error-passive node will not affect any other node on the network, unless the bus error is detected by the node that is actually transmitting (i.e. the bus master). If the bus master node generates an error-passive flag, this may cause other nodes to generate Error Frames due to the resulting bit-stuffing violation. After transmission of an Error Frame an error-passive node must wait for 6 consecutive recessive bits on the bus before attempting to rejoin bus communications.

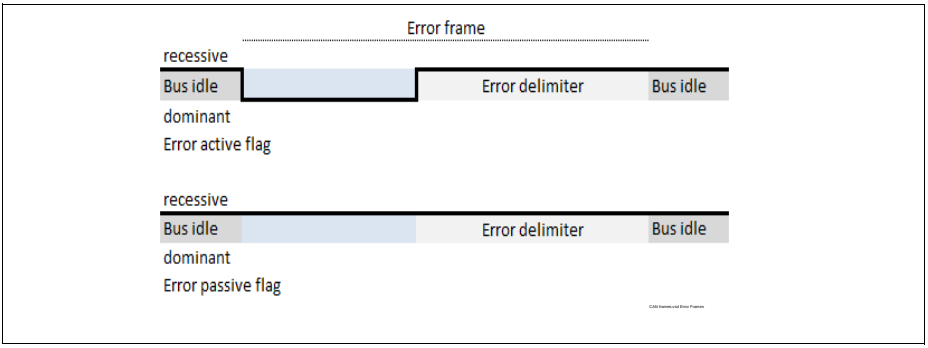


Figure 21-5 CAN Error Frames

A CAN FD node operating in the Data-Phase will switch back to the Arbitration Phase when starting an Error Flag.

21.1.3.4 Overload Frame

The Overload Frame consists of two bit fields which are Overload Flag and Overload Delimiter.

Overload conditions

If a CAN FD node samples a dominant bit at the eight bit (last bit) of an Error Delimiter or Overload Delimiter, or if a CAN FD Receiver samples a dominant bit at the last bit of End of Frame, it will start transmitting an Overload Frame (not an Error Frame). The Error Counters will not be incremented.

21.1.4 The Nominal Bit Time

One bit cell (this means one high or low pulse of the NRZ code) is composed by four segments. Each segment is an integer multiple of Time Quanta t_Q. The Time Quanta is the smallest discrete timing resolution used by a CAN node. The nominal bit time definition with its segments is shown in **Figure 21-6**.

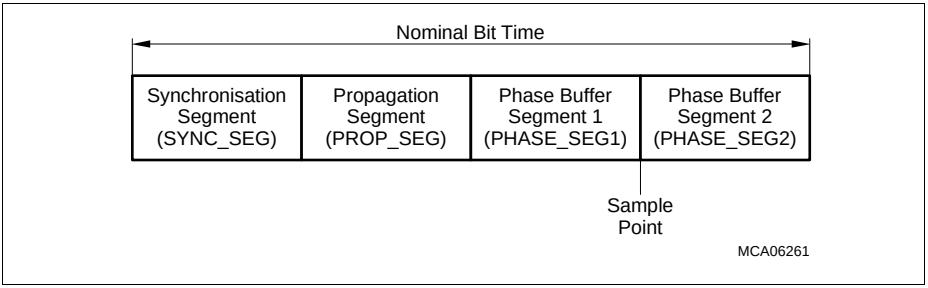


Figure 21-6 Partition of Nominal Bit Time

The Synchronization Segment (SYNC_SEG) is used to synchronize the various bus nodes. If there is a bit state change between the previous bit and the current bit, then the bus state change is expected to occur within this segment. The length of this segment is always 1 t_Q.

The Propagation Segment (PROP_SEG) is used to compensate for signal delays across the network. These delays are caused by signal propagation delay on the bus line and through the electronic interface circuits of the bus nodes.

The Phase Segments 1 and 2 (PHASE_SEG1, PHASE_SEG2) are used to compensate for edge phase errors. These segments can be lengthened or shortened by re-synchronization. PHASE_SEG2 is reserved for calculation of the subsequent bit level, and is ≥ 2 t_Q. At the sample point, the bus level is read and interpreted as the value of the bit cell. It occurs at the end of PHASE_SEG1.

The total number of t_Q in a bit time is between 8 and 25.

As a result of re-synchronization, PHASE_SEG1 can be lengthened or PHASE_SEG2 can be shortened. The amount of lengthening or shortening the phase buffer segments has an upper limit given by the re-synchronization jump width. The re-synchronization jump width may be between 1 and 4 t_Q, but it may not be longer than PHASE_SEG1.

21.1.4.1 CAN FD bit timing

The first part of a CAN FD frame until the BRS bit is transmitted with the Nominal Bit Rate. The bit rate is switched if the BRS bit is recessive, until the CRC Delimiter is reached or until the CAN FD controller sees an error condition that results in the starting

of an Error Frame. CAN FD Error Frames, as well as ACK Field, End of Frame, Overload Frames and all in CAN Format are transmitted with the Nominal Bit Rate

Synchronization Rules

Hard synchronization and Resynchronization are the two forms of synchronization. They obey the following rules,

- Hard synchronization is performed whenever there is a recessive to dominant edge during Bus Idle, Suspend Transmission, and second or third bits of Intermission. Hard synchronization is also performed at the recessive to the dominant edge from FDF to r0 in CAN FD format frames
- A transmitter shall not resynchronize while it transmits in the CAN FD data phase

21.1.5 Error Detection and Error Handling

The CAN protocol has sophisticated error detection mechanisms. The following errors can be detected:

- **Cyclic Redundancy Check (CRC) Error**
  With the CRC, the transmitter calculates special check bits for the bit sequence from the start of a frame until the end of the Data Field. This CRC sequence is transmitted in the CRC Field. The receiving node also calculates the CRC sequence using the same formula, and performs a comparison to the received sequence. If a mismatch is detected, a CRC error has occurred and an Error Frame is generated. The message is repeated.
- **Acknowledge Error**
  In the Acknowledge Field of a message, the transmitter checks whether a dominant bit is read during the Acknowledge Slot (that is sent out as a recessive bit). If not, no other node has received the frame correctly, an Acknowledge Error has occurred, and the message must be repeated. No Error Frame is generated.
- **Form Error**
  If a transmitter detects a dominant bit in one of the four segments End of Frame, Interframe Space, Acknowledge Delimiter, or CRC Delimiter, a Form Error has occurred, and an Error Frame is generated. The message is repeated.
- **Bit Error**
  A Bit Error occurs if a) a transmitter sends a dominant bit and detects a recessive bit or b) if the transmitter sends a recessive bit and detects a dominant bit when monitoring the actual bus level and comparing it to the just transmitted bit. In case b), no error occurs during the Arbitration Field (ID, RTR, IDE) and the Acknowledge Slot.
- **Stuff Error**
  If between Start of Frame and CRC Delimiter, six consecutive bits with the same polarity are detected, the bit-stuffing rule has been violated. A stuff error occurs and an Error Frame is generated. The message is repeated.

Detected errors are made public to all other nodes via Error Frames (except Acknowledge Errors). The transmission of the erroneous message is aborted and the frame is repeated as soon as possible. Furthermore, each CAN node is in one of the three error states (error-active, error-passive or bus-off) according to the value of the internal error counters. The error-active state is the usual state where the bus node can transmit messages and active-error frames (made of dominant bits) without any restrictions. In the error-passive state, messages and passive-error frames (made of recessive bits) may be transmitted. The bus-off state makes it temporarily impossible for the node to participate in the bus communication. During this state, messages can be neither received nor transmitted.

Basic CAN, Full CAN

There is one more CAN characteristic that is related to the interface of a CAN module (controller) and the host CPU: Basic-CAN and Full-CAN functionality.

In Basic-CAN devices, only basic functions of the protocol are implemented in hardware, such as the generation and the check of the bit stream. The decision, whether a received message has to be stored or not (acceptance filtering), and the complete message management must be done by software.

Full-CAN devices (this is the case for the MultiCAN+ controller as implemented in TC21x/TC22x/TC23x) manage the whole bus protocol in hardware, including the acceptance filtering and message management. Full-CAN devices contain message objects that handle autonomously the identifier, the data, the direction (receive or transmit) and the information of CAN operation. During the initialization of the device, the host CPU determines which messages are to be sent and which are to be received. The host CPU is informed by interrupt if the identifier of a received message matches with one of the programmed (receive-) message objects. The CPU load of Full-CAN devices is greatly reduced. When using Full-CAN devices, high baud rates and high bus loads with many messages can be handled.

21.2 Overview

The MultiCAN+ module provides a communication interface which is fully compliant with CAN specification V2.0B (active) and to CAN FD ISO11898-1 DIS version 2014, providing communications at up to 1 Mbit/s in Classical CAN (ISO 11898-1:2003(E) mode and/or CAN FD until 5 MBaud dataspeed (dependent on frequency and nodes)).

The MultiCAN+ module for the TC21x/TC22x/TC23x consists of 2[1)] module (i.e MultiCAN with 3 CAN nodes and MultiCAN1 with 3 CAN Nodes), representing 6[2)] serial communication interfaces. . Each CAN node communicates over two pins (TXD and RXD). The device ports which are used for TXD and RXD may be individually configured within the PORTS block. Several port configuration options are available to provide application-specific flexibility.

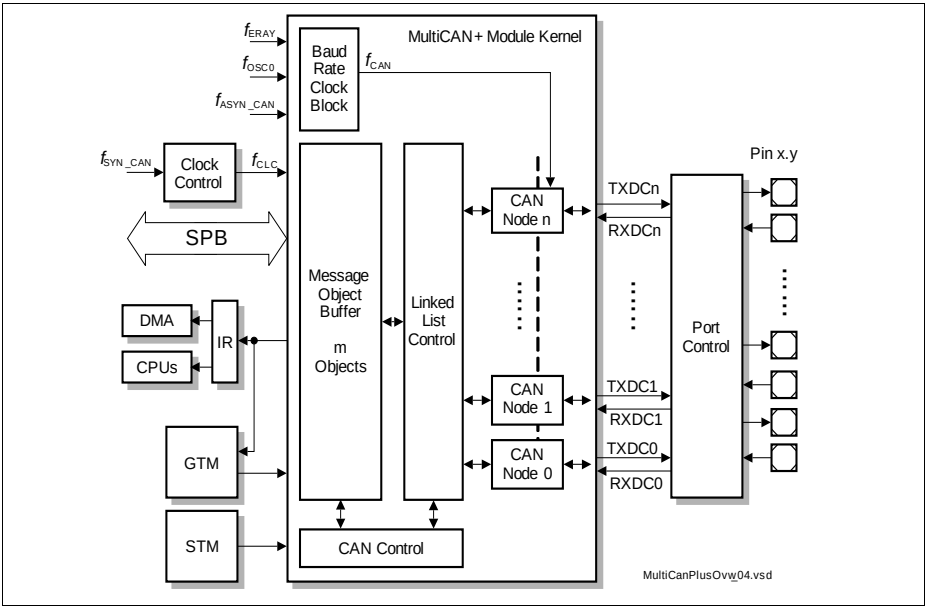


Figure 21-7 Overview of the MultiCAN+ Module with 3 nodes and 128 message objects in AURIX products.

The MultiCAN+ module contains 3/3 independently operating CAN nodes with Full-CAN functionality that are able to exchange Data and Remote Frames via a gateway function.

1) TC23x consists of 2 modules (MultiCAN and MultiCAN1), TC22x consists of 1module (MultiCAN).

2) TC23x has 6 CAN serial communication interfaces, TC22x has 3 CAN communication interfaces.

Each CAN node can receive and transmit standard frames with 11-bit identifiers as well as extended frames with 29-bit identifiers.

All CAN nodes share a common set of 128/128 message objects. Each message object can be individually allocated to one of the CAN nodes.Besides serving as a storage container for incoming and outgoing frames, message objects can be combined to build gateways between the CAN nodes or to setup a FIFO buffer.

The message objects are organized in double-chained linked lists, where each CAN node has its own list of message objects. A CAN node stores frames only into message objects that are allocated to the message object list of the CAN node, and it transmits only messages belonging to this message object list. A powerful, command-driven list controller performs all message object list operations.

The bit timings for the CAN nodes are derived from the module timer clock (f_{CAN}) and are programmable up to a data rate of 1 Mbit/s in Classical CAN (ISO 11898-1:2003(E) mode or up to 5 MBaud in CAN FD mode. External bus transceivers are connected to a CAN node via a pair of receive and transmit pins.

21.2.1 Features List

The MultiCAN+ module provides the following features:

- Compliant with ISO 11898 and SAE J 1939
- CAN functionality according to CAN specification V2.0 B active
- Supports CAN with Flexible Data-Rate Specification CAN FD DIS version ISO11898-1 2014 with max. 64 data bytes[1)]
- Debug over CAN[2)]
- Dedicated control registers for each CAN node
- Data transfer rates up to 1 Mbit/swhen operating in Classical CAN mode per ISO 11898-1:2003(E)
- Supports up to 5MBaud, when operating in CAN FD mode.
- Support for asynchronous clock sources for baud rate generation by providing separate frequency domain and input:
  - System frequency clock f_{CLC}
  - Low-jitter E-Ray PLL clock
  - Direct oscillator clock (e.g. from ceramic resonator)
- Frequency jitter calibration based on external CAN messages during runtime
- Flexible and powerful message transfer control and error handling capabilities
- Advanced CAN bus bit timing analysis and baud rate detection for each CAN node via a frame counter
- Full-CAN functionality: A set of 128/128[3)] message objects can be individually

1) At time of release of this document ISO 11898-1 (Classical CAN and CAN FD) is in working draft mode, an errata sheet will document differences when ISO 11898-1 has been finalized.

2) Debug over CAN feature is available through CAN module only and not available on CAN1 module.

  - Allocated (assigned) to any CAN node
  - Configured as transmit or receive object
  - Setup to handle frames with 11-bit or 29-bit identifier
  - Identified by a timestamp via a frame counter
  - Configured to remote monitoring mode
- Advanced Acceptance Filtering
  - Each message object provides an individual acceptance mask to filter incoming frames
  - A message object can be configured to accept standard or extended frames or to accept both standard and extended frames
  - Message objects can be grouped into different priority classes for transmission and reception
  - The selection of the message to be transmitted first can be based on frame identifier, IDE bit and RTR bit according to CAN arbitration rules, or on its order in the list
- Advanced CAN node features
  - Analyzer Mode supports monitoring of bus traffic without actively participating on the bus
  - Internal Loop-Back Mode is available for test purposes
  - An internal timer is provided that can detect a receive or remote frame time-out; the message objects to be supervised are selectable
  - Data transmission from a node can be stopped without affecting reception
  - Programmable minimum delay between two consecutive messages
- Advanced message object functionality
  - Message objects can be combined to build FIFO message buffers of arbitrary size, limited only by the total number of message objects
  - Message objects can be linked to form a gateway that automatically transfers frames between 2 different CAN buses. A single gateway can link any two CAN nodes. An arbitrary number of gateways can be defined
- Advanced data management
  - The message objects are organized in double-chained lists
  - List reorganizations can be performed at any time, even during full operation of the CAN nodes
  - A powerful, command-driven list controller manages the organization of the list structure and ensures consistency of the list
  - Message FIFOs are based on the list structure and can easily be scaled in size during CAN operation
- Advanced interrupt handling
  - Message interrupts, node interrupts can be generated

3) MultiCAN has 128 Message Objects, MultiCAN1 has 128 Message Objects.

  - Interrupt requests can be routed individually to one of the 16 or 8 interrupt output lines
  - Message post-processing notifications can be combined flexibly into a dedicated register field of 256 notification bits
- Module internal SRAM with ECC protection
- Pretended Networking Operation for energy saving purposes
  - If direct oscillator clock is selected for baud rate generation, CPU clock can be slowed down or CPU Idle Mode can be entered
  - Three selectable message object can be transmitted periodically without CPU involvement, triggered by a CAN node timer or by STM or GTM modules.
  - Message objects are received without CPU involvement and can wake up the system via receive interrupt
  - AUTOSAR optimized and backward compatible with existing MultiCAN software

21.3 CAN Flexible Data-Rate (CAN FD)

CAN Flexible Data-Rate (CAN FD) builds on existing CAN (ISO 11898-1) specifications allowing higher data rates and larger payloads. This is achieved with a new CAN FD frame format different from existing Classical CAN format, both frame formats can coexist within the same network. Classical CAN nodes and CAN FD nodes can communicate with each other as long as CAN FD frame format is not being used.

CAN FD functionality is available on all nodes of MultiCAN+ module.

Feature Lists

- Supports CAN Flexible Data-Rate Specification V1.0

21.3.1 Transmitter Delay Compensation

The CAN protocol requires that transmitted data be compared with the received data from its local CAN transceiver to determine if there are any transmit errors. In the case of CAN FD data phase when a faster bit rate is used, resulting in shorter bit timing, the delay caused by the local transmitting loop delay will be greater than TSEG1 (time segment before sample point) causing a bit error to be detected. The transmitter loop delay limits the bit rate in the data phase of a CAN FD frame.

Thus to overcome this limitation, a Transmitter Delay Compensation feature is introduced where a new sample point (Secondary Sample Point) shall be used by transmitters in the data phase of a CAN FD frame, the sample point which does not account for the transceiver loop delay is ignored.

The Secondary Sample Point consists of the transmitter loop delay and a configurable Transmitter Delay Compensation offset (NTDCRx.TDCO). (i.e The secondary sample point is reached by counting the total delay consisting of the compensation offset and measured transceiver loop delay.)

The Transmitter loop delay is measured in each transmitted frame at the edge from the FDF bit to the following bit r0, between the edge of the transmitted bit and the edge of the received bit. (see **Figure 21-8**) The count down is started with the begin of the bit time (transmit point).

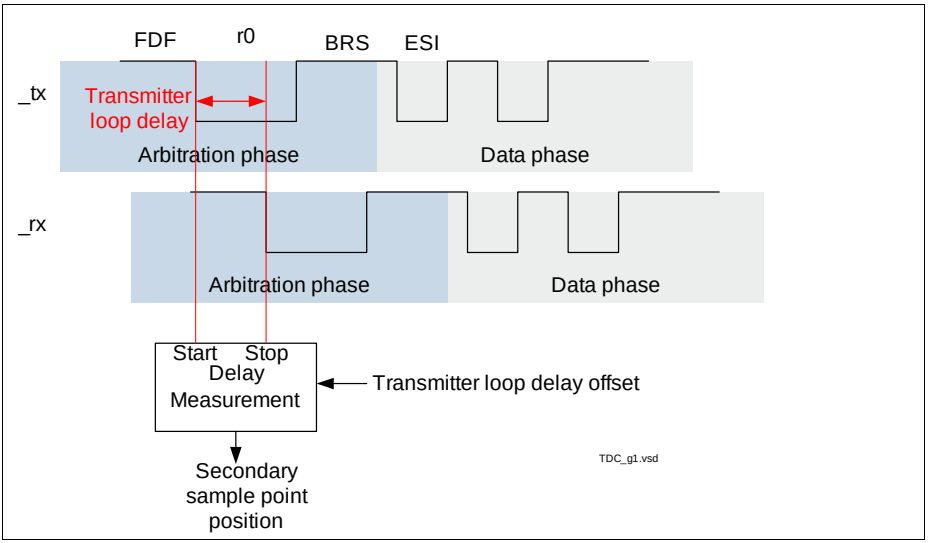


Figure 21-8 Transmitter Delay Loop measurement

Transmitter Delay compensation offset (NTDCRx.TDCO) is used to adjust the secondary sample point inside the bit time (e.g half of the bit time in the data phase). Finally the resulting Secondary Sample Point is rounded down to the next integer number of time quanta t_q and placed after the end of the transmitted bit. If a bit error is detected at the Secondary Sample Point, the transmitter will react to this bit error at the next sample point.

(i.e When a bit error is detected by the Transmitter Delay compensation unit, the error is reported at the next sample point and becomes visible (in error active case) at the transmit point that follows the regular sample point.)

Secondary Sample Point is used in the Data Phase of CAN FD and Sample Point is used in the Arbitration Phase of CAN FD. The Transmitter Delay compensation which determines the secondary sample point is able to handle a total delay (measured transceiver loop delay + compensation offset) which goes beyond the current bit time.

MultiCAN+ module allows the secondary sample point to be placed anywhere within the current and next 3 bit times (i.e covers up to 4 data phase bit rate). The maximum delay which can be compensated by MultiCAN+ delay compensation during the data phase is 4 bit times.

Note:

7. *CAN receive input line contributes to the measured loop delay.*
8. *Measurement granularity of the Transmitter Delay compensation is the fast time quantum given by the fast baud rate prescaler.*

21.4 MultiCAN+ Kernel Functional Description

This section describes the functionality of the MultiCAN+ module.

21.4.1 Module Structure

Figure 21-9 shows the general structure of the MultiCAN+ module.

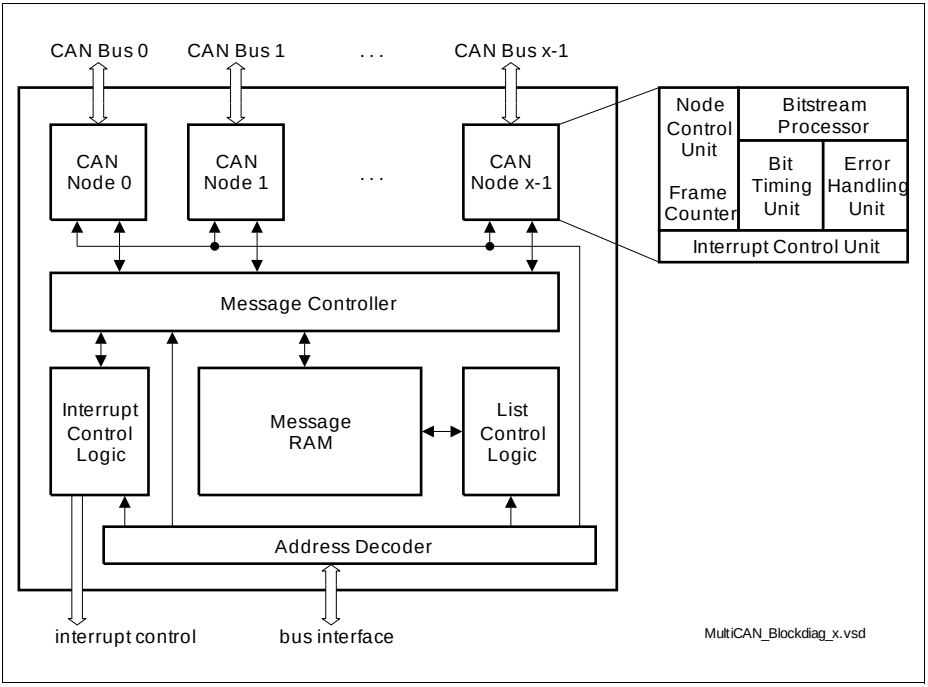


Figure 21-9 MultiCAN+ Block Diagram

CAN Nodes

Each CAN node consists of several sub-units.

- **Bitstream Processor**
  The Bitstream Processor performs data, remote, error and overload frame processing according to the ISO 11898 standard. This includes conversion between the serial data stream and the input/output registers.
- **Bit Timing Unit**
  The Bit Timing Unit determines the length of a bit time and the location of the sample point according to the user settings, taking into account propagation delays and phase shift errors. The Bit Timing Unit also performs resynchronization.

- **Error Handling Unit**
  The Error Handling Unit manages the receive and transmit error counter. Depending on the contents of both counters, the CAN node is set into an error-active, error passive or bus-off state.
- **Node Control Unit**
  The Node Control Unit coordinates the operation of the CAN node:
  – Enable/disable CAN transfer of the node
  – Enable/disable and generate node-specific events that lead to an interrupt request (CAN bus errors, successful frame transfers etc.)
  – Administration of the frame counter and of the node timers
- **Interrupt Control Unit**
  The Interrupt Control Unit in the CAN node controls the interrupt generation for the different conditions that can occur in the CAN node.

Message Controller

The Message Controller handles the exchange of CAN frames between the CAN nodes and the message objects that are stored in the Message RAM. The Message Controller performs several functions:

- Receive acceptance filtering to determine the correct message object for storing of a received CAN frame
- Transmit acceptance filtering to determine the message object to be transmitted first, individually for each CAN node
- Transfer contents between message objects and the CAN nodes, taking into account the status/control bits of the message objects
- Handling of the FIFO buffering and gateway functionality
- Aggregation of message-pending notification bits

List Controller

The List Controller performs all operations that lead to a modification of the double-chained message object lists. Only the list controller is allowed to modify the list structure. The allocation/deallocation or reallocation of a message object can be requested via a user command interface (command panel). The list controller state machine then performs the requested command autonomously.

Interrupt Control

The general interrupt structure is shown in **Figure 21-10**. The interrupt event can trigger the interrupt generation. The interrupt pulse is generated independently of the interrupt flag in the interrupt status register. The interrupt flag can be reset by software by writing a 0 to it.

If enabled by the related interrupt enable bit in the interrupt enable register, an interrupt pulse can be generated at one of the 16 or 8 interrupt output lines INT_Om of the MultiCAN+ module. If more than one interrupt source is connected to the same interrupt node pointer (in the interrupt node pointer register), the requests are combined to one common line.

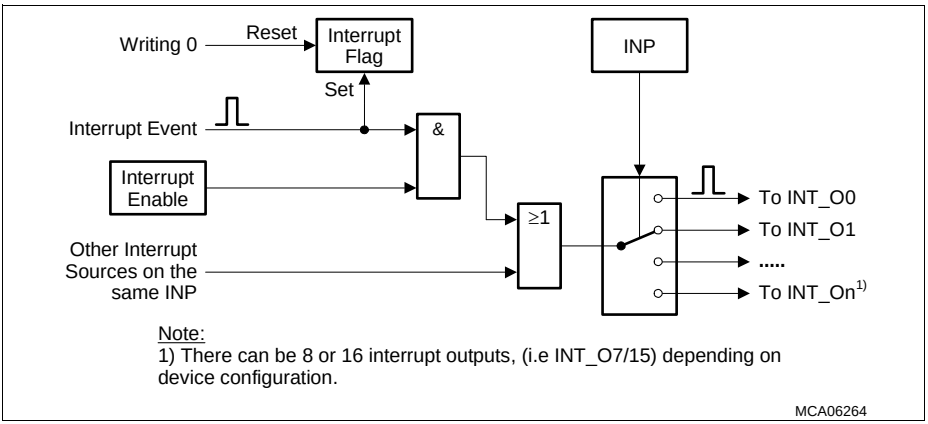


Figure 21-10 General Interrupt Structure

21.4.2 Clock Control

The CAN module timer clock f_{CAN} of the functional blocks of the MultiCAN+ module is derived from the asynchronous, higher precision clock f_A or from the synchronous clock source. The Fractional Divider is used to generate f_{CAN} used for bit timing calculation, The frequency of f_{CAN} is identical for all CAN nodes. The register file operate with the module control clock f_{CLC}. See also **“MultiCAN+ Clock Generation” on Page 21-21**.

The output clock f_{CAN} of the Fractional Divider is based on the clock f_A, but only every nth clock pulse is taken. The suspend signal (coming as acknowledge from the MultiCAN+ module in response to a OCDS suspend request) freezes or resets the Fractional Divider. The clock select and fractional divider are reset together with the CAN module kernel when the module kernel reset is triggered.

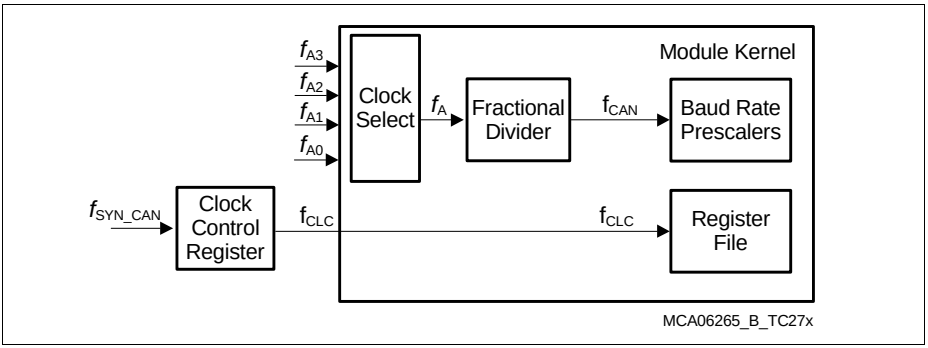


Figure 21-11 MultiCAN+ Clock Generation

The f_{SYN_CAN} is identical to f_{SPB}. f_{Ai} is the asynchronous clock input.

Table 21-2 indicates the minimum operating frequencies in MHz for f_{CLC} that are required for a baud rate of 1 Mbit/s for the active CAN nodes. If a lower baud rate is desired, the values can be scaled linearly (e.g. for a maximum of 500 kbit/s, 50% of the indicated value are required).

For CAN FD operations, please refer to **Table 21-3** for the minimum operating frequency in MHz for f_{CLC} that are required.

The values imply that the CPU (or DMA) executes maximum accesses to the MultiCAN+ module. The values may contain rounding effects.

Table 21-2 Minimum Operating Frequencies[1)2)] [MHz]

Number of allocated message objects MO[3)],	Number of Active CAN Nodes				
	1	**2**	**3**	**4**	**5**
16 MO	12	19	26	33	40
32 MO	15	23	30	37	44
64 MO	21	28	37	46	53
128 MO	40	45	50	55	61
256 MO	72	77	82	88	93

1) In the case of 15 time quanta, the minimum operating frequency required is 15MHz.

2) To guarantee the minimum operating frequencies when low power mode (pretended networking mode) is enabled (i.e when frequency of f_{SPB} is reduced), f_{SYN_CAN} and f_{CLC} are switched from f_{SPB} to f_{CAN} when SCU register CCUCON.LPDIV>0.

3) Only those message objects have to be taken into account that are allocated to a CAN node. The unallocated message objects have no influence on the minimum operating frequency.

Table 21-3 Minimum Operating Frequencies for CAN FD [MHz]

No. of allocated MO[1)]	No. of Active CAN Nodes	Acceleration Factor[2)3)]				
		1	2	3	4	5
16 MO	1	20	40	60	80	100
	2	39	48	72	96	-
	3	58	58	84	-	-
	4	77	77	96	-	-
	5	96	96	-	-	-
32 MO	1	20	40	60	80	100
	2	39	48	72	96	-
	3	58	58	84	-	-
	4	77	77	96	-	-
	5	96	96	-	-	-
64 MO	1	27	40	60	80	100
	2	39	51	72	96	-
	3	58	65	84	-	-
	4	77	79	96	-	-
	5	96	96	-	-	-
128 MO	1	47	65	75	81	100
	2	57	79	91	99	-
	3	67	93	-	-	-
	4	77	-	-	-	-
	5	96	-	-	-	-

1) Only those message objects have to be taken into account that are allocated to a CAN node. The unallocated message objects have no influence on the minimum operating frequency.
When message objects are configured as transmit or receive FIFO structures i.e MOFCRn.MMC = 0001/0010, only the base object of 1 is counted towards the minimum frequency requirement, all other slave objects on the FIFO do not count towards the minimum frequency requirement. See **Section 21.4.12.5**

2) Acceleration Factor is the ratio of the Data Bit Rate to Nominal Bit Rate. The Nominal Bit Rate is taken as 1Mbit/s in table above. As an example, an acceleration factor(A.F) of 4 refers to a Data Bit Rate of 4Mbit/s and Nominal Bit Rate is 1Mbit/s.
Only the node operating with the highest baudrate need to be considered when several nodes operate at different acceleration factor. e.g When 1node operate at A.Fof 3 and 3 nodes operate at A.F of 2, the minimum operating frequency required is determined by the node operate at A.F of 3.

3) Please note that other combinations of acceleration factor are possible, values here are for illustrative purposes only. Values with dash '-' indicates required f_{CLC} exceeding the maximum frequency capability off product SCU typically at 100MHz.

The baud rate generation of the MultiCAN+ being based on f_A, this frequency has to be chosen carefully to allow correct CAN bit timing. The required value of f_A is given by an integer multiple (n) of the CAN baud rate multiplied by the number of time quanta per CAN bit time. For example, to reach 1 Mbit/s with 20 tq per bit time, possible values of f_A are given by formula [n × 20] MHz, with n being an integer value, starting at 1.

In order to minimize jitter, it is not recommended to use the fractional divider mode for high baud rates.

Additionally, for correct operation of the MultiCAN, the following conditions have to be fulfilled,

$$\text{Baudrate}_{max} = [(8 \times T_{CAN}) + (8 \times T_{CLC}) + (4 \times \text{No. of active CAN nodes} \times T_{CLC})] \quad (21.1)$$

also

$$\text{NBTR.SJW} < \text{NBTR.TSEG1}$$

As an example, when f_{CLC} = 10MHz, f_{CAN} = 20MHz, No of active CAN nodes =2,

Baudrate_{max}=[(8x50ns) + (8x100ns) + (4x2x100ns)] = 2000ns = 500 KBaud

Table 21-4 below illustrates the minimum CAN module timer clock f_{CAN} and Module Control Clock f_{CLC} that's required to support a baudrate generation of 500KBaud. If a higher baudrate is desired, the values need to be calculated as per **Equation (21.1)**.

Table 21-4 Minimum Operating Frequencies [MHz] required for 500KBaud

No. of active CAN nodes	$f_{CAN} = f_{CLC}$ (MHz)	$f_{CAN} \neq f_{CLC}$ (MHz)[1]	
		f_{CAN}	f_{CLC}
1	10	16	8
		20	8
		24	8
		80	7
2	12	16	11
		20	10
		24	10
		80	9
3	14	16	14
		20	13
		24	12
		80	11

1) To guarantee the minimum operating frequencies when low power mode (pretended networking mode) is enabled (i.e when frequency of f_{SPB} is reduced), f_{SYN_CAN} and f_{CLC} are switched from f_{SPB} to f_{CAN} when SCU register CCUCON.LPDIV>0.

21.4.3 Port Input Control

It is possible to select the input lines for the RXDCANx inputs for the CAN nodes. The selected input is connected to the CAN node and is also available to wake-up the system. More details are defined in **Section 21.7.5.2** on **Page 21-170**.

21.4.4 OCDS Suspend

The OCDS suspend of MultiCAN+ is controlled with the **OCS** register. MultiCAN+ module provides the following Suspend Modes:

Hard Suspend Mode (VERSION: RW ACCESS NEEDS KERNEL CLOCKING) (All actions are immediately stopped)

The Hard Suspend mode is not to be used in normal CAN applications, this mode is meant for debugging the peripheral IP within a controlled environment.

In Hard Suspend Mode, the MultiCAN+ module clocks f_{CLC} and f_{CAN} are switched off. Reading and writing of registers is possible but will enable the kernel clock for a few cycles. In this mode, there is a very high probability that the communication with other CAN devices is made impossible and that the CAN bus is blocked (e.g. if the suspended CAN module just sends a dominant level). A reset operation must be executed to leave Hard Suspend Mode.

Attention: Register accesses with clocking in Hard Suspend Mode can have unintended side effects like signals becoming and staying active. This can affect also other modules, so a MultiCAN+ kernel reset might not be sufficient to bring the system into a defined state.

Soft Suspend Mode (The current action is finished)

In Soft Suspend Mode, the MultiCAN+ module clock f_{CLC} is kept running. Module functions are stopped automatically after internal actions have been finished before it enters the suspended state with **OCS**.SUSSTA set (for example, after a CAN frame has been sent out). Due to this behavior, the communication network is not blocked. All registers are accessible for read and write actions. As a result, the debugger can stop the module actions and modify registers. These modifications are taken into account after the Suspend Mode is left. Soft Suspend mode does not influence the kernel clock.

The Hard Suspend Mode can be enabled/disabled only for the complete MultiCAN+ module. The Soft Suspend Mode can be individually enabled for each CAN node.

The user has to be aware that the Soft Suspend Mode can corrupt the TTCAN timing values, because the counter clock can be disabled. In order to guarantee correct TTCAN

behavior, a TTCAN node should not enter any Suspend Mode if the consistency of the timing data must be ensured.

Note: During suspend mode, kernel registers may be accessed only when the kernel clock is running.

21.4.5 OCDS Trigger Bus (OTGB) Interface

The MultiCAN Trigger Set is shown in **Table 21-5**. Its output is on OTGB0 or OTGB1 controlled by the **OCS** register.

Table 21-5 TS16_CAN Trigger Set MultiCAN

Bits	Name	Description
[7:0]	MON	Message Object Number for transmit and receive direction. Only valid when TT or RT is active and FMI is inactive.
[11:8]	NN	Node Number for transmit and receive direction. Only valid when TT or RT is active and FMI is inactive.
12	TT	Transmit Trigger. Active for one clock cycle when the transmit message is setup.
13	RT	Receive Trigger. Active for one clock cycle after End-of-Frame (EOF) processing for a receive message.
14	FMI	Foreign Message Indicator. The received message will not be copied to any message object. Only valid when RT is active.
15		Reserved

MCDS Trigger Setup

Table 21-6 lists the qualification methods for TC16_CAN with MCDS. The methods are not exclusive and can be combined.

Table 21-6 TS16_CAN MCDS Trigger Setup

Qualification	Method
Transmit and/or receive messages	Positive edge sensitivity on TS16_CAN.TT and/or RT. Configured in the OTGM MCDS I/F.
No foreign messages.	MCDS data value condition TS16_CAN < 4000_H
Specific message object	MCDS masked data value comparison. TS16_CAN & $00FF_H$ == message object number
Specific node	MCDS masked data value comparison. TS16_CAN & $0F00_H$ == (node number) << 8

21.4.6 CAN Node Control

Each CAN node may be configured and run independently of the other CAN node. Each CAN node is equipped with its own node control logic to configure the global behavior and to provide status information.

Note: In the following descriptions, index “x” stands for the node number and index “n” represents the message object number.

Configuration Mode is activated when bit NCRx.CCE is set to 1. This mode allows CAN bit timing parameters and the error counter registers to be modified.

CAN Analyzer Mode

CAN Analyzer Mode is activated when bit NCRx.CALM is set to 1. In this operation mode, Data And Remote Frames are monitored without active participation in any CAN transfer (CAN transmit pin is held on recessive level). Incoming Remote Frames are stored in a corresponding transmit message object, while arriving data frames are saved in a matching receive message object.

In CAN Analyzer Mode, the entire configuration information of the received frame is stored in the corresponding message object, and can be evaluated by the CPU to determine their identifier, IDE bit information and data length code (ID and DLC optionally if the Remote Monitoring Mode is active, bit MOFCRn.RMM = 1). Incoming frames are not acknowledged, and no Error Frames are generated. If CAN Analyzer Mode is enabled, Remote Frames are not responded to by the corresponding Data Frame, and Data Frames cannot be transmitted by setting the transmit request bit MOSTATn.TXRQ. Receive interrupts are generated in CAN Analyzer Mode (if enabled) for all error free received frames.

The node-specific interrupt configuration is also defined by the Node Control Logic via the NCRx register bits TRIE, ALIE and LECIE:

- If control bit TRIE is set to 1, a transfer interrupt is generated when the NSRx register has been updated (after each successfully completed message transfer).
- If control bit ALIE is set to 1, an alert interrupt is generated when a “bus-off” condition has been recognized or the Error Warning Level has been exceeded or under-run. Additionally, list or object errors lead to this type of interrupt.
- If control bit LECIE is set to 1, a last error code interrupt is generated when an error code > 0 is written into bit field NSRx.LEC by hardware.

Setting bit TXDIS in register NCRx stops the transmit activity of this node without affecting reception; bit CANDIS disables the node completely.

The Node x Status Register NSRx provides an overview about the current state of the respective CAN node x, comprising information about CAN transfers, CAN node status, and error conditions.

The CAN frame counter can be used to check the transfer sequence of message objects or to obtain information about the instant a frame has been transmitted or received from the associated CAN bus. CAN frame counting is performed by a 16-bit counter, controlled by register NFCRx. Bit fields NFCRx.CFMOD and NFCRx.CFSEL determine the operation mode and the trigger event incrementing the frame counter.

21.4.6.1 Bit Timing Unit

According to the ISO 11898 standard, a CAN bit time is subdivided into different segments (**Figure 21-12**). Each segment consists of multiples of a time quantum t_q. The magnitude of t_q is adjusted by Node x Bit Timing Register bit fields NBTRx.BRP and NBTRx.DIV8, both controlling the baud rate prescaler (register NBTRx is described on **Page 21-106**). The baud rate prescaler is driven by the module timer clock f_{CAN} (generation and control of f_{CAN} is described on **Page 21-165**).

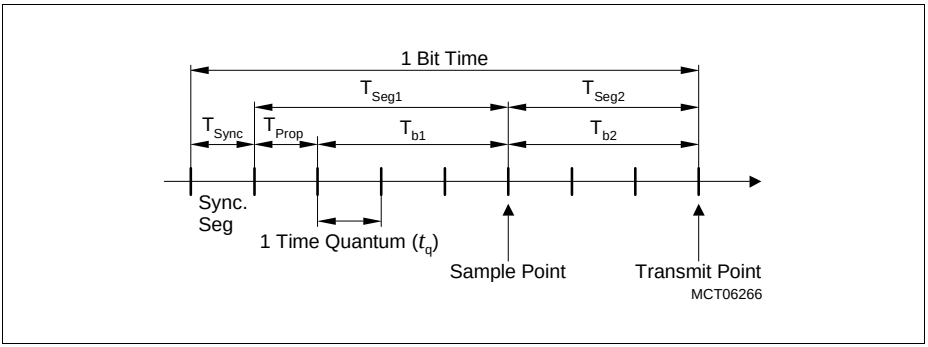


Figure 21-12 CAN Bus Bit Timing Standard

The Synchronization Segment (T_{Sync}) allows a phase synchronization between transmitter and receiver time base. The Synchronization Segment length is always one t_q. The Propagation Time Segment (T_{Prop}) takes into account the physical propagation delay in the transmitter output driver on the CAN bus line and in the transceiver circuit. For a working collision detection mechanism, T_{Prop} must be two times the sum of all propagation delay quantities rounded up to a multiple of t_q. The phase buffer segments 1 and 2 (T_{b1}, T_{b2}) before and after the signal sample point are used to compensate for a mismatch between transmitter and receiver clock phases detected in the synchronization segment.

The maximum number of time quanta allowed for re-synchronization is defined by bit field NBTRx.SJW. The Propagation Time Segment and the Phase Buffer Segment 1 are combined to parameter T_{Seg1}, which is defined by the value NBTRx.TSEG1. A minimum of 3 time quanta is demanded by the ISO standard. Parameter T_{Seg2}, which is defined by the value of NBTRx.TSEG2, covers the Phase Buffer Segment 2. A minimum of 2 time

quanta is demanded by the ISO standard. According to ISO standard, a CAN bit time, calculated as the sum of T_{Sync}, T_{Seg1} and T_{Seg2}, must not fall below 8 time quanta.

Calculation of the bit time:

t_q	$= (BRP + 1) / f_{CAN}$	if DIV8 = 0
	$= 8 \times (BRP + 1) / f_{CAN}$	if DIV8 = 1
T_{Sync}	$= 1 \times t_q$	
T_{Seg1}	$= (TSEG1 + 1) \times t_q$	(min. 3 t_q)
T_{Seg2}	$= (TSEG2 + 1) \times t_q$	(min. 2 t_q)
bit time	$= T_{Sync} + T_{Seg1} + T_{Seg2}$	(min. 8 t_q)

To compensate phase shifts between clocks of different CAN controllers, the CAN controller must synchronize on any edge from the recessive to the dominant bus level. The hard synchronization is enabled (at the start of frame), the bit time is restarted at the synchronization segment. Otherwise, the re-synchronization jump width T_{SJW} defines the maximum number of time quanta, a bit time may be shortened or lengthened by one re-synchronization. The value of SJW is defined by bit field NBTRx.SJW.

T_{SJW}	$= (SJW + 1) \times t_q$
T_{Seg1}	$\geq T_{SJW} + T_{prop}$
T_{Seg2}	$\geq T_{SJW}$

The maximum relative tolerance for f_{CAN} in classical CAN format and CAN FD format depends on the Phase Buffer Segments, re-synchronization jump width and the bit time.

Classical CAN Format

df_{CAN}	$\leq \min (T_{b1}, T_{b2}) / [2 \times (13 \times \text{bit time} - T_{b2})]$	AND
df_{CAN}	$\leq T_{SJW} / 20 \times \text{bit time}$	

CAN FD Format

df_{CAN}	$\leq \min (T_{b1(N)}, T_{b2(N)}) / [2 \times (13 \times \text{bit time}_{(N)} - T_{b2(N)})]$	AND
df_{CAN}	$\leq T_{SJW(N)} / 20 \times \text{bit time}_{(N)}$	AND
df_{CAN}	$\leq T_{SJW(D)} / 20 \times \text{bit time}_{(D)}$	AND

$df_{CAN} \leq \min (T_{b1(D)}, T_{b2(D)}) / (2 \times [(6 \times \text{bit time}_{(D)} - T_{b2(D)}) \times BRP_{(D)} / BRP_{(N)} + (7 \times \text{bit time}_{(N)})] ($ AND

$df_{CAN} \leq [T_{SJW(D)} - (BRP_{(N)} / BRP_{(D)} - 1)] / (2 \times [(2 \times \text{bit time}_{(N)} - T_{b2(N)}) \times BRP_{(N)} / BRP_{(D)} + T_{b2(D)} + 4 \times \text{bit time}_{(D)}])$

A valid CAN bit timing must be written to the CAN Node Bit Timing Register NBTR and Fast Node Bit Timing Register before resetting the INIT bit in the Node Control Register, i.e. before enabling the operation of the CAN node.

The Node Bit Timing Register may be written only if bit CCE (Configuration Change Enable) is set in the corresponding Node Control Register.

21.4.6.2 Bitstream Processor

Based on the message objects in the message buffer, the Bitstream Processor generates the remote and Data Frames to be transmitted via the CAN bus. It controls the CRC generator and adds the checksum information to the new remote or Data Frame. After including the SOF bit and the EOF field, the Bitstream Processor starts the CAN bus arbitration procedure and continues with the frame transmission when the bus was found in idle state. While the data transmission is running, the Bitstream Processor continuously monitors the I/O line. If (outside the CAN bus arbitration phase or the acknowledge slot) a mismatch is detected between the voltage level on the I/O line and the logic state of the bit currently sent out by the transmit shift register, a CAN LEC error interrupt request is generated, and the error code is indicated by the Node x Status Register bit field NSRx.LEC.

The data consistency of an incoming frame is verified by checking the associated CRC field. When an error has been detected, a CAN LEC error interrupt request is generated and the associated error code is presented in the Node x Status Register NSRx. Furthermore, an Error Frame is generated and transmitted on the CAN bus. After decomposing a faultless frame into identifier and data portion, the received information is transferred to the message buffer executing remote and Data Frame handling, interrupt generation and status processing.

21.4.6.3 Error Handling Unit

The Error Handling Unit of a CAN node x is responsible for the fault confinement of the CAN device. Its two counters, the Receive Error Counter REC and the Transmit Error Counter TEC (bit fields of the Node x Error Counter Register NECNTx, see **Page 21-113**) are incremented and decremented by commands from the Bitstream Processor. If the Bitstream Processor itself detects an error while a transmit operation is running, the Transmit Error Counter is incremented by 8. An increment of 1 is used when the error condition was reported by an external CAN node via an Error Frame generation. For error analysis, the transfer direction of the disturbed message and the

node that recognizes the transfer error are indicated for the respective CAN node x in register NECNTx. Depending on the values of the error counters, the CAN node is set into error- active, error-passive, or bus-off state.

The CAN node is in error-active state if both error counters are below the error-passive limit of 128. The CAN node is in error-passive state, if at least one of the error counters is equal to or greater than 128.

The bus-off state is activated if the Transmit Error Counter is equal to or greater than the bus-off limit of 256. This state is reported for CAN node x by the Node x Status Register flag NSRx.BOFF. The device remains in this state, until the “bus-off” recovery sequence is finished. Additionally, Node x Status Register flag NSRx.EWRN is set when at least one of the error counters is equal to or greater than the error warning limit defined by the Node x Error Count Register bit field NECNTx.EWRNLVL. Bit NSRx.EWRN is reset if both error counters fall below the error warning limit again (see **Page 21-96**).

21.4.6.4 CAN Frame Counter

Each CAN node is equipped with a frame counter that counts transmitted/received CAN frames or obtains information about the time when a frame has been started to transmit or be received by the CAN node. CAN frame counting/bit time counting is performed by a 16-bit counter that is controlled by Node x Frame Counter Register NFCRx (see **Page 21-114**). Bit field NFCRx.CFSEL determines the operation mode of the frame counter:

- **Frame Count Mode:**
  After the successful transmission and/or reception of a CAN frame, the frame counter is copied into the CFCVAL bit field of the MOIPRn register of the message object involved in the transfer. Afterwards, the frame counter is incremented.
- **Time Stamp Mode:**
  The frame counter is incremented with the beginning of a new bit time. When the transmission/reception of a frame starts, the value of the frame counter is captured and stored to the CFC bit field of the NFCRx register. After the successful transfer of the frame the captured value is copied to the CFCVAL bit field of the MOIPRn register of the message object involved in the transfer.
- **Bit Timing Mode:**
  Used for baud rate detection and analysis of the bit timing (**Chapter 21.4.8.3**).
- **Error Count Mode:**
  The frame counter is incremented when an error frame is received or an error is detected by the node (010_B to 110_B) (see **Table 21-12** for **Encoding of the LEC Bit field**). If the NFCRx.CFCIE interrupt bit is enabled, the NFCRx.CFCOV overflow flag will be set when the frame counter overflows.

21.4.6.5 Node Timing Functions

A CAN node offers the following timing functions:

- A receive time-out mode that can detect reception of message objects; the message objects to be supervised are selectable.
- Without CPU involvement, a selectable message object can be transmitted periodically, triggered by a timer, a System Timer (STM) or General Timer Module (GTM).

The clocking options for the node timer is controlled by Node Timer Clock Control Register, **CAN_NTCCRx (x = 0-2)**, the node timing functions for Receive Timeout Mode is controlled by Node x Timer Receive Timeout Register, **CAN_NTRTRx (x = 0-2)** and for Transmit Trigger Mode is controlled by Node x Timer A/B/C Transmit Trigger Registers, **CAN_NTATTRx (x = 0-2)**, **CAN_NTBTTRx (x = 0-2)**, **CAN_NTCTTRx (x = 0-2)**.

Modes with Timer Usage

A CAN node timer is driven by the CAN bit time clock, divided by a prescaler selected via bit field TPSC in the corresponding timer control register. The timers are enabled by writing to the RELOAD bits in the relevant node timer registers; then it decrements from its initial value. The further behavior depends on the selected timer mode:

- **Receive Timeout Mode:**
  By setting bit MOFCRn.RXTOE, a receive time-out check is enabled for message object n, which may be a receive or remote data object. If any of the participating message objects is received before the timer is 0, the timer will be reloaded. When the timer reaches 0, it will stop. Bit TE in NTRTRx register will be set. With bit TEIE = 1, a node interrupt will be generated.
- **Transmit Trigger Mode:**
  When the timer reaches 0, Bit MOSTATn.TXRQ of the message object n selected by bit field TXMO in the timer control register will be set.

Transmit Trigger by System Timer or General Timer Module

For Node x Timer the trigger for transmission of a message can also be set by a System Timer (STM) trigger event or General Timer Module (GTM) trigger event. See **Figure 21-13**.
Bit NTCCRx.TRIGSRC in the timer clock control register enables this feature and the timer is started once values are written to the RELOAD bits of the relevant NTATTx, NTBTTx, or NTCTTx registers. In transmit trigger mode, when a trigger event occurs (STM or GTM), the node timer will be decremented per trigger event timing prescaled by (TPSC+1) till it reaches zero where Bit MOSTATn.TXRQ of the message object n selected by bit field TXMO in the timer control register will be set.

Note: The transmit request bits (i.e NTATTx.TXMO / NTBTTx.TXMO / NTCTTx.TXMO) in the timer control register of a node is able to trigger transmit request (MOSTATn.TXRQ) of message object in a list belonging to any other CAN nodes.

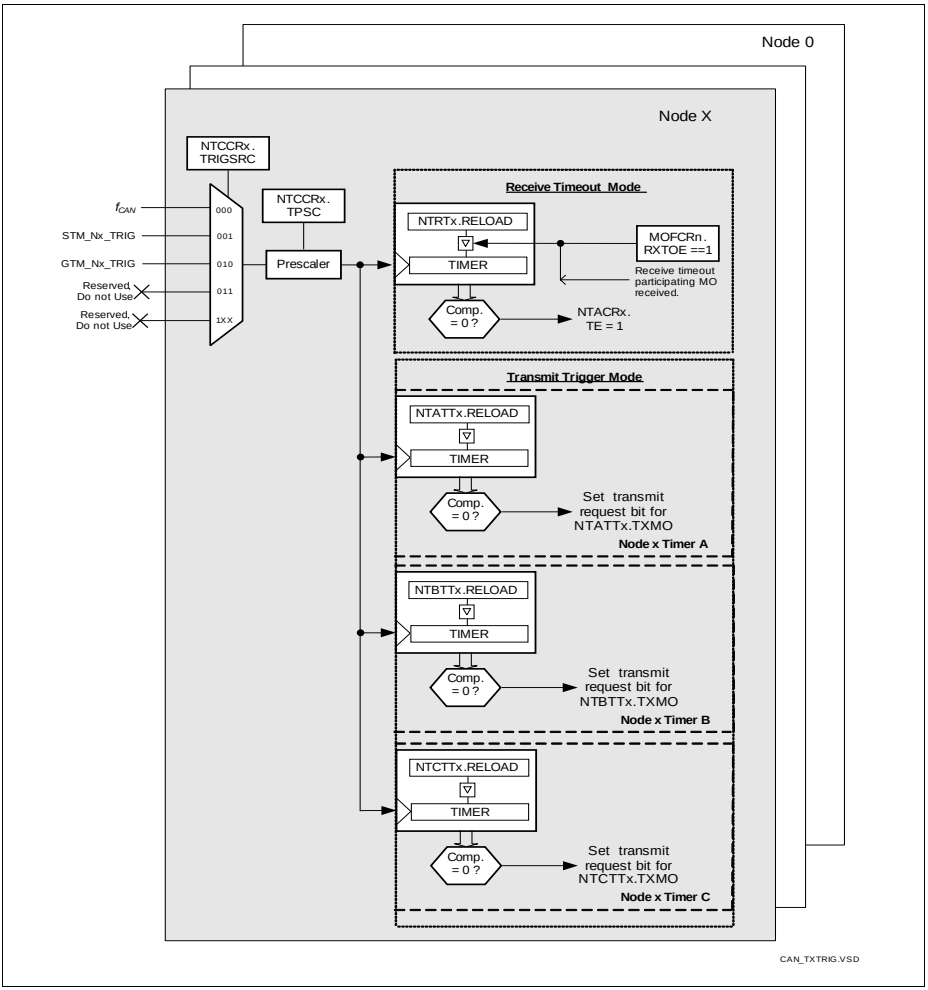


Figure 21-13 CAN Node Timing Modes

21.4.6.6 CAN Node Interrupts

Each CAN node has four five hardware triggered interrupt request types that are able to generate an interrupt request upon:

- The successful transmission or reception of a frame
- A CAN protocol error with a last error code
- An alert condition: Transmit/receive error counters reach the warning limit, bus-off state changes, a List Length Error occurs, or a List Object Error occurs
- An overflow of the frame counter
- A node timer A, C or D event

Besides the hardware generated interrupts, software initiated interrupts can be generated using the Module Interrupt Trigger Register MITR. Writing a 1 to bit n of bit field MITR.IT generates an interrupt request signal on the corresponding interrupt output line INT_On. When writing MITR.IT more than one bit can be set resulting in activation of multiple INT_On interrupt output lines at the same time. See also **“Interrupt Control” on Page 21-174** for further processing of the CAN node interrupts.

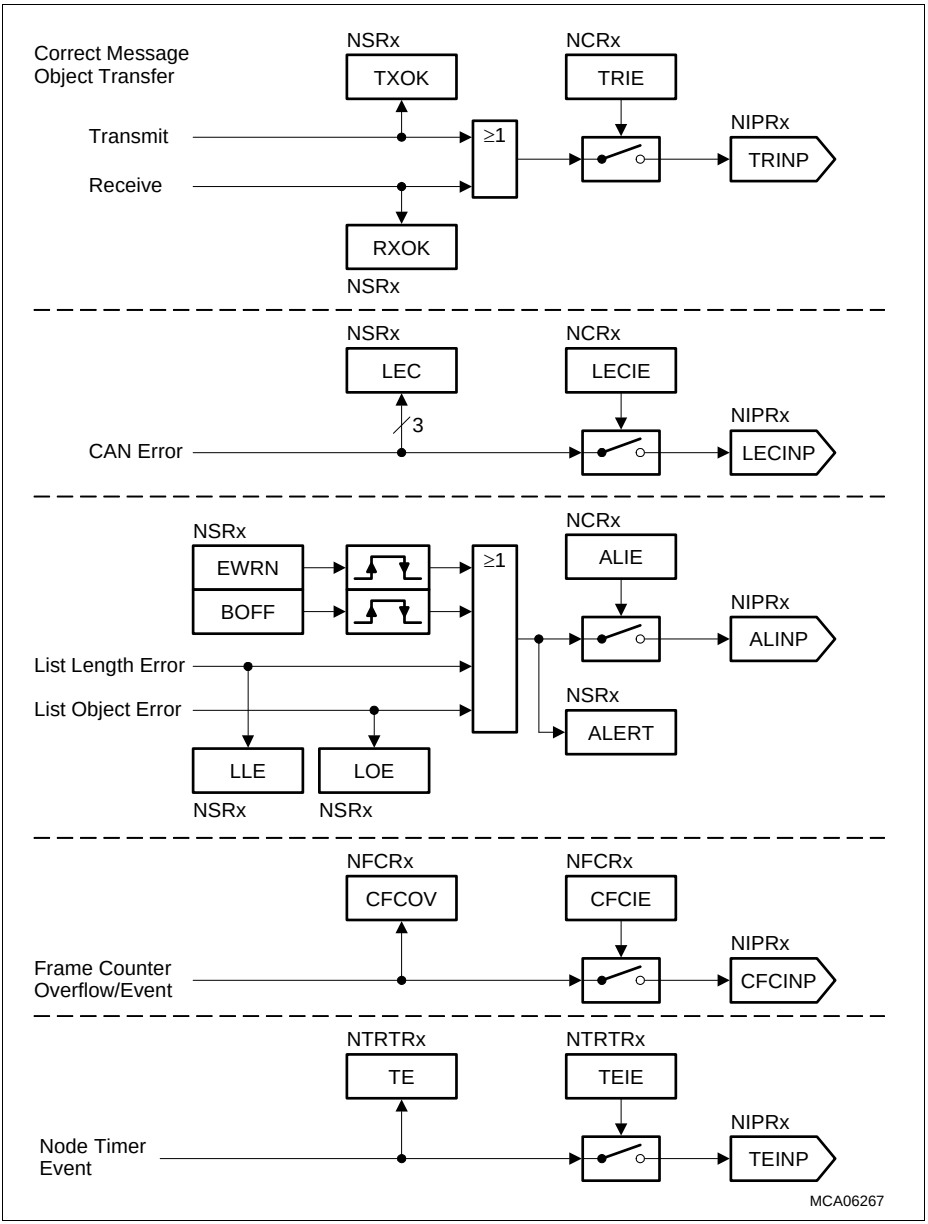


Figure 21-14 CAN Node Interrupts

21.4.7 Message Object List Structure

This section describes the structure of the message object lists in the MultiCAN+ module.

21.4.7.1 Basics

The message objects of the MultiCAN+ module are organized in double-chained lists, where each message object has a pointer to the previous message object in the list as well as a pointer to the next message object in the list. The MultiCAN+ module provides 16 lists. Each message object is allocated to one of these lists. In the example in **Figure 21-15**, the three message objects (3, 5, and 16) are allocated to the list with index 2 (List Register LIST2).

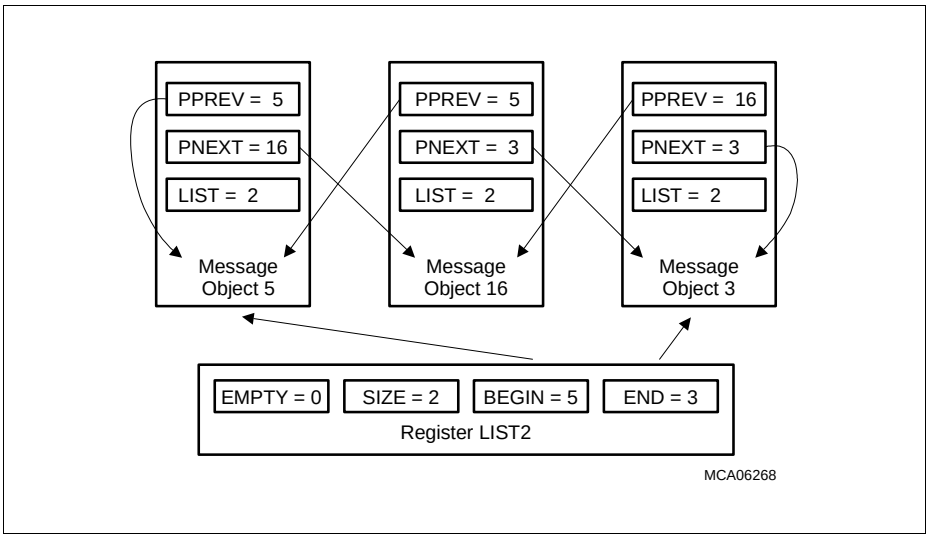


Figure 21-15 Example Allocation of Message Objects to a List

Bit field BEGIN in the List Register (for definition, see **Page 21-87**) points to the first element in the list (object 5 in the example), and bit field END points to the last element in the list (object 3 in the example). The number of elements in the list is indicated by bit field SIZE of the List Register (SIZE = number of list elements - 1, thus SIZE = 2 for the 3 elements in the example). The EMPTY bit of the List Register indicates whether or not a list is empty (EMPTY = 0 in the example, because list 2 is not empty).

Each message object n has a pointer PNEXT in its Message Object n Status Register MOSTATn (see **Page 21-128**) that points to the next message object in the list, and a pointer PPREV that points to the previous message object in the list. PPREV of the first message object points to the message object itself because the first message object has

no predecessor (in the example message object 5 is the first message object in the list, indicated by PPREV = 5). PNEXT of the last message object also points to the message object itself because the last message object has no successor (in the example, object 3 is the last message object in the list, indicated by PNEXT = 3).

Bit field MOSTATn.LIST indicates the list index number to which the message object is currently allocated. The message object of the example are allocated to list 2. Therefore, all LIST bit fields for the message objects assigned to list 2 are set to LIST = 2.

21.4.7.2 List of Unallocated Elements

The list with list index 0 has a special meaning: it is the list of all unallocated elements. An element is called unallocated if it belongs to list 0 (MOSTATn.LIST = 0). It is called allocated if it belongs to a list with an index not equal to 0 (MOSTATn.LIST > 0).

After reset, all message objects are unallocated. This means that they are assigned to the list of unallocated elements with MOSTATn.LIST = 0. After this initial allocation of the message objects caused by reset, the list of all unallocated message objects is ordered by message number (predecessor of message object n is object n-1, successor of object n is object n+1).

21.4.7.3 Connection to the CAN Nodes

Each CAN node is linked to one unique list of message objects. A CAN node performs message transfer only with the message objects that are allocated to the list of the CAN node. This is illustrated in **Figure 21-16**. Frames that are received on a CAN node may only be stored in one of the message objects that belongs to the CAN node; frames to be transmitted on a CAN node are selected only from the message objects that are allocated to that node, as indicated by the vertical arrows.

There are more lists (16) than CAN nodes (3). This means that some lists are not linked to one of the CAN nodes. A message object that is allocated to one of these unlinked lists cannot receive messages directly from a CAN node and it may not transmit messages.

FIFO and gateway mechanisms refer to message numbers and not directly to a specific list. The user must take care that the message objects targeted by FIFO/gateway belong to the desired list. The mechanisms make it possible to work with lists that do not belong to the CAN node.

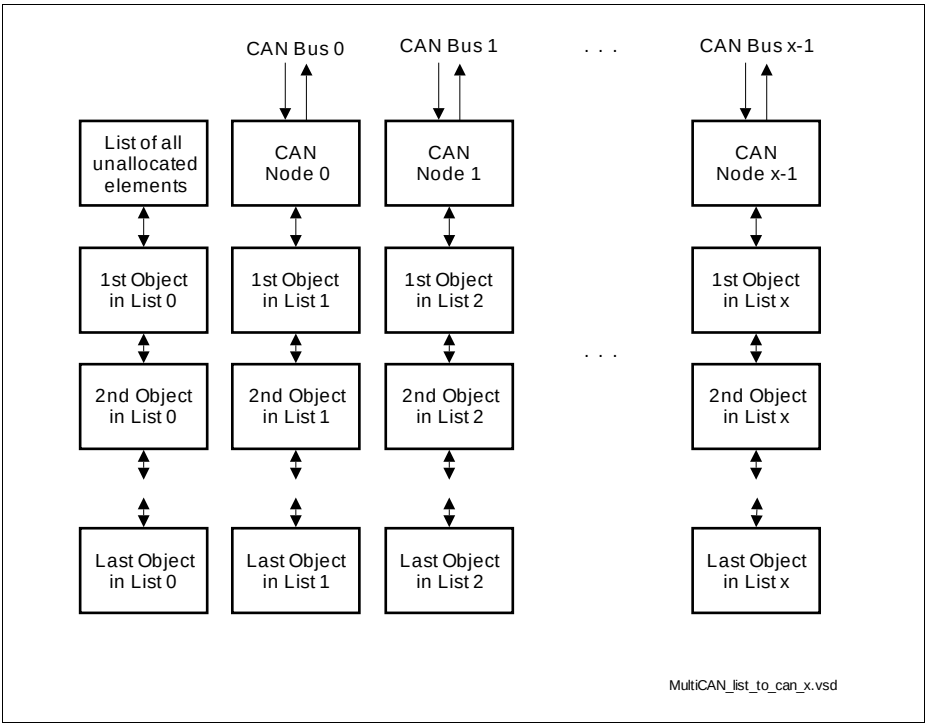


Figure 21-16 Message Objects Linked to CAN Nodes

21.4.7.4 List Command Panel

The list structure cannot be modified directly by write accesses to the LIST registers and the PPREV, PNEXT and LIST bit fields in the Message Object Status Registers, as they are read only. The list structure is managed by and limited to the list controller inside the MultiCAN+ module. The list controller is controlled via a command panel allowing the user to issue list allocation commands to the list controller. The list controller has two main purposes:

1. Ensure that all operations that modify the list structure result in a consistent list structure.
2. Present maximum ease of use and flexibility to the user.

The list controller and the associated command panel allows the programmer to concentrate on the final properties of the list, which are characterized by the allocation of message objects to a CAN node, and the ordering relation between objects that are allocated to the same list. The process of list (re-)building is done in the list controller.

Table 21-7 gives an overview on the available panel commands while Table 21-11 on Page 21-81 describes the panel commands in more detail.

Table 21-7 Panel Commands Overview

Command Name	Description
No Operation	No new command is started.
Initialize Lists	Run the initialization sequence to reset the CTRL and LIST field of all message objects.
Static Allocate	Allocate message object to a list.
Dynamic Allocate	Allocate the first message object of the list of unallocated objects to the selected list.
Static Insert Before	Remove a message object (source object) from the list that it currently belongs to, and insert it before a given destination object into the list structure of the destination object.
Dynamic Insert Before	Insert a new message object before a given destination object.
Static Insert Behind	Remove a message object (source object) from the list that it currently belongs to, and insert it behind a given destination object into the list structure of the destination object.
Dynamic Insert Behind	Insert a new message object behind a given destination object.

A panel command is started by writing the respective command code into the Panel Control Register bit field PANCTR.PANCMD (see Page 21-80). The corresponding command arguments must be written into bit fields PANCTR.PANAR1 and PANCTR.PANAR2 before writing the command code, or latest along with the command code in a single 32-bit write access to the Panel Control Register.

With the write operation of a valid command code, the PANCTR.BUSY flag is set and further write accesses to the Panel Control Register are ignored. The BUSY flag remains active and the control panel remains locked until the execution of the requested command has been completed. After a reset and resetting the CLC.DISR (see Page 21-167) register, the list controller builds up list 0. Afterwards the BUSY bit is set, dependent on core speed this might be visible. During list controller initialization, BUSY is set and other accesses to the CAN RAM are forbidden. The CAN RAM can be accessed again when BUSY becomes inactive.

Note: The CAN RAM is automatically initialized after reset by the list controller in order to ensure correct list pointers in each message object. The end of this CAN RAM initialization is indicated by bit PANCTR.BUSY becoming inactive.

In case of a dynamic allocation command that takes an element from the list of unallocated objects, the PANCTR.RBUSY bit is also set along with the BUSY bit (RBUSY = BUSY = 1). This indicates that bit fields PANAR1 and PANAR2 are going to be updated by the list controller in the following way:

1. The message number of the message object taken from the list of unallocated elements is written to PANAR1.
2. If ERR (bit 7 of PANAR2) is set to 1, the list of unallocated elements was empty and the command is aborted. If ERR is 0, the list was not empty and the command will be performed successfully.

The results of a dynamic allocation command are written before the list controller starts the actual allocation process. As soon as the results are available, RBUSY becomes inactive (RBUSY = 0) again, while BUSY still remains active until completion of the command. This allows the user to set up the new message object while it is still in the process of list allocation. The access to message objects is not limited during ongoing list operations. However, any access to a register resource located inside the RAM delays the ongoing allocation process by one access cycle.

As soon as the command is finished, the BUSY flag becomes inactive (BUSY = 0) and write accesses to the Panel Control Register are enabled again. Also, the “No Operation” command code is automatically written to the PANCTR.PANCMD field. A new command may be started any time when BUSY = 0.

All fields of the Panel Control Register PANCTR except BUSY and RBUSY may be written by the user. This makes it possible to save and restore the Panel Control Register if the Command Panel is used within independent (mutually interruptible) interrupt service routines. If this is the case, any task that uses the Command Panel and that may interrupt another task that also uses the Command Panel should poll the BUSY flag until it becomes inactive and save the whole PANCTR register to a memory location before issuing a command. At the end of the interrupt service routine, the task should restore PANCTR from the memory location.

Before a message object that is allocated to the list of an active CAN node shall be moved to another list or to another position within the same list, bit MOSTATn.MSGVAL (“Message Valid”) of message object n must be cleared.

21.4.8 CAN Node Analyzer Mode

The chapter describes the CAN node analyzer capabilities of the MultiCAN+ module.

21.4.8.1 Analyzer Mode

The CAN Analyzer Mode makes it possible to monitor the CAN traffic for each CAN node individually without affecting the logical state of the CAN bus. The CAN Analyzer Mode for CAN node x is selected by setting Node x Control Register bit NCRx.CALM.

In CAN Analyzer Mode, the transmit pin of a CAN node is held at a recessive level permanently. The CAN node may receive frames (Data, Remote, and Error Frames) but is not allowed to transmit. Received Data/Remote Frames are not acknowledged (i.e. acknowledge slot is sent recessive) but will be received and stored in matching message objects as long as there is any other node that acknowledges the frame. The complete message object functionality is available, but no transmit request will be executed. CAN Analyzer mode works for both Classical CAN format and CAN FD format.

21.4.8.2 Loop-Back Mode

The MultiCAN+ module provides a Loop-Back Mode to enable an in-system test of the MultiCAN+ module as well as the development of CAN driver software without access to an external CAN bus.

The loop-back feature consists of an internal CAN bus (inside the MultiCAN+ module) and a bus select switch for each CAN node (see **Figure 21-17**). With the switch, each CAN node can be connected either to the internal CAN bus (Loop-Back Mode activated) or the external CAN bus, respectively to transmit and receive pins (normal operation). The CAN bus that is not currently selected is driven recessive; this means the transmit pin is held at 1, and the receive pin is ignored by the CAN nodes that are in Loop-Back Mode.

The Loop-Back Mode is selected for CAN node x by setting the Node x Port Control Register bit NPCRx.LBM. All CAN nodes that are in Loop-Back Mode may communicate together via the internal CAN bus without affecting the normal operation of the other CAN nodes that are not in Loop-Back Mode.

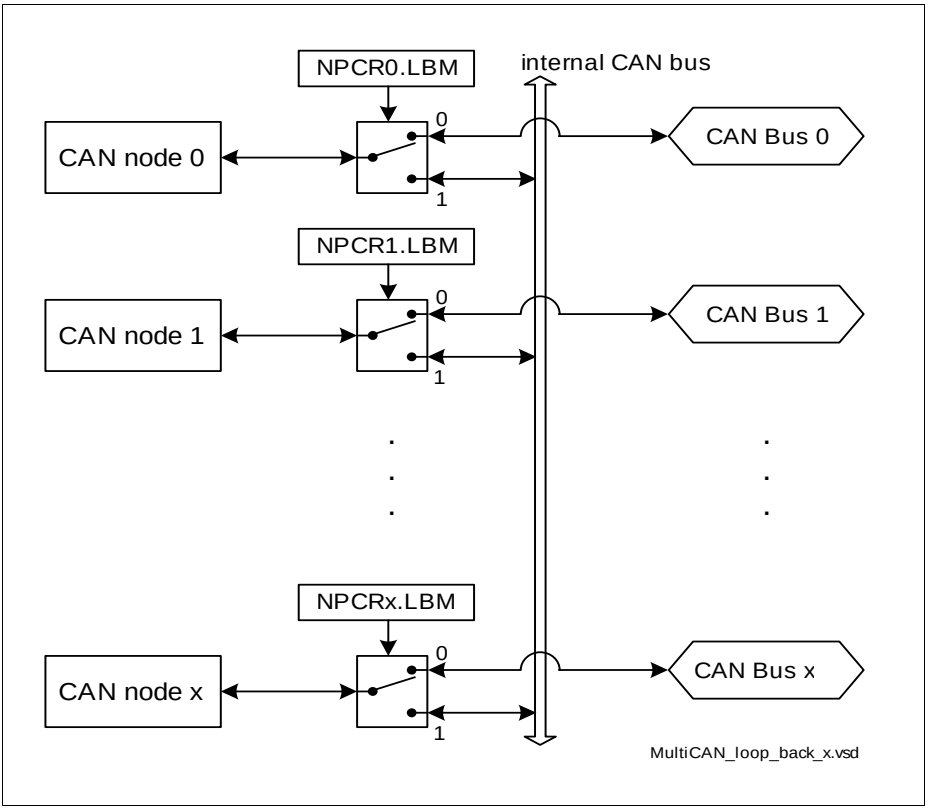


Figure 21-17 Loop-Back Mode

21.4.8.3 Bit Timing Analysis

Detailed analysis of the bit timing can be performed for each CAN node using the analysis modes of the CAN frame counter. The bit timing analysis functionality of the frame counter may be used for automatic detection of the CAN baud rate, as well as to analyze the timing of the CAN network.

Bit timing analysis for CAN node x is selected when bit field NFCRx.CFMOD = 10_B. Bit timing analysis does not affect the operation of the CAN node.

The bit timing measurement results are written into the NFCRx.CFC bit field. Whenever NFCRx.CFC is updated in bit timing analysis mode, bit NFCRx.CFCOV is also set to indicate the CFC update event. The value of NFCRx.CFC is valid one module cycle later when NFCRx.CFCOV is set. If NFCRx.CFCIE is set, an interrupt request can be generated (see **Figure 21-14**).

Automatic Baud Rate Detection

For automatic baud rate detection, the time between the observation of subsequent dominant edges on the CAN bus must be measured. This measurement is automatically performed if bit field NFCRx.CFSEL = 000_B. With each dominant edge monitored on the CAN receive input line, the time (measured in f_{CAN} clock cycles) between this edge and the most recent dominant edge is stored in the NFCRx.CFC bit field.

Synchronization Analysis

The bit time synchronization is monitored if NFCRx.CFSEL = 010_B. The time between the first dominant edge and the sample point is measured and stored in the NFCRx.CFC bit field. The bit timing synchronization offset may be derived from this time as the first edge after the sample point triggers synchronization and there is only one synchronization between consecutive sample points.

Synchronization analysis can be used, for example, for fine tuning of the baud rate during reception of the first CAN frame with the measured baud rate.

Driver Delay Measurement

The delay between a transmitted edge and the corresponding received edge is measured when NFCRx.CFSEL = 011_B (dominant to dominant) and NFCRx.CFSEL = 100_B (recessive to recessive). These delays indicate the time needed to represent a new bit value on the physical implementation of the CAN bus.

21.4.9 Message Acceptance Filtering

The chapter describes the Message Acceptance Filtering capabilities of the MultiCAN+ module.

21.4.9.1 Receive Acceptance Filtering

When a CAN frame is received by a CAN node, a unique message object is determined in which the received frame is stored after successful frame reception. A message object is qualified for reception of a frame if the following six conditions are met.

- The message object is allocated to the message object list of the CAN node by which the frame is received.
- Bit MOSTATn.MSGVAL in the Message Object Status Register (see **Page 21-128**) is set.
- Bit MOSTATn.RXEN is set.
- Bit MOSTATn.DIR is equal to bit RTR of the received frame.
  If bit MOSTATn.DIR = 1 (transmit object), the message object accepts only Remote Frames. If bit MOSTATn.DIR = 0 (receive object), the message object accepts only Data Frames.

- If bit MOAMRn.MIDE = 1, the IDE bit of the received frame becomes evaluated in the following way: If MOARn.IDE = 1, the IDE bit of the received frame must be set (indicates extended identifier). If MOARn.IDE = 0, the IDE bit of the received frame must be cleared (indicates standard identifier).
  If bit MOAMRn.MIDE = 0, the IDE bit of the received frame is “don’t care”. In this case, message objects with standard and extended frames are accepted.
- The identifier of the received frame matches the identifier stored in the Arbitration Register of the message object as qualified by the acceptance mask in the MOAMRn register. This means that each bit of the received message object identifier is equal to the bit field MOARn.ID, except those bits for which the corresponding acceptance mask bits in bit field MOAMRn.AM are cleared. These identifier bits are “don’t care” for reception. **Figure 21-18** illustrates this receive message identifier check.

Among all messages that fulfill all six qualifying criteria the message object with the highest receive priority wins receive acceptance filtering and becomes selected to store the received frame. All other message objects lose receive acceptance filtering.
The following priority scheme is defined for the message objects:
A message object a (MOa) has higher receive priority than a message object b (MOb) if the following two conditions are fulfilled (see **Page 21-145**):

1. MOa has a higher priority class than MOb. This means, the 2-bit priority bit field MOARa.PRI must be equal or less than bit field MOARb.PRI.
2. If both message objects have the same priority class (MOARa.PRI = MOARb.PRI), MOb is a list successor of MOa. This means that MOb can be reached by means of successively stepping forward in the list, starting from a.

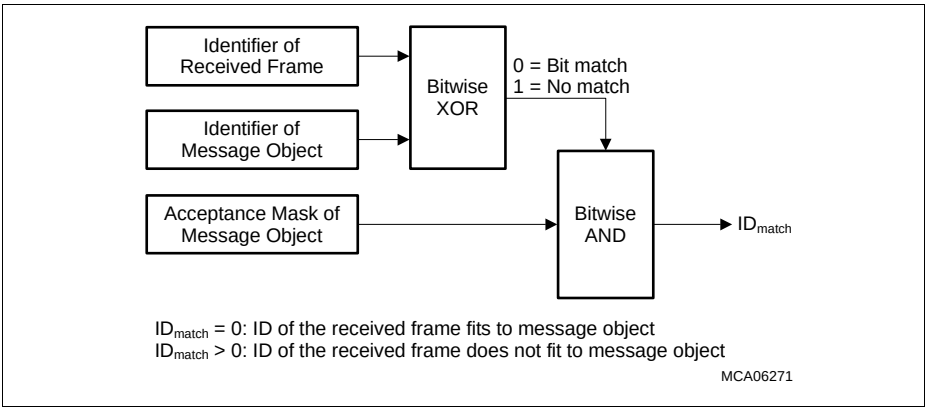


Figure 21-18 Received Message Identifier Acceptance Check

21.4.9.2 Transmit Acceptance Filtering

A message is requested for transmission by setting a transmit request in the message object that holds the message. If more than one message object have a valid transmit request for the same CAN node, one of these message objects is chosen for transmission, because only a single message object can be transmitted at one time on a CAN bus.

A message object is qualified for transmission on a CAN node if the following four conditions are met (see also **Figure 21-19**).

1. The message object is allocated to the message object list of the CAN node.
2. Bit MOSTATn.MSGVAL is set.
3. Bit MOSTATn.TXRQ is set.
4. Bit MOSTATn.TXEN0 and MOSTATn.TXEN1 are set.

A priority scheme determines which one of all qualifying message objects is transmitted first. It is assumed that message object a (MOa) and message object b (MOb) are two message objects qualified for transmission. MOb is a list successor of MOa. For both message objects, CAN messages CANa and CANb are defined (identifier, IDE, and RTR are taken from the message-specific bit fields and bits MOARn.ID, MOARn.IDE and MOSTATn.DIR).
If both message objects belong to the same priority class (identical PRI bit field in register MOARn), MOa has a higher transmit priority than MOb if one of the following conditions is fulfilled.

- PRI = 10_B and CAN message MOa has higher or equal priority than CAN message MOb with respect to CAN arbitration rules (see **Table 21-19** on **Page 21-146**).
- PRI = 01_B or PRI = 11_B (priority by list order).

The message object that is qualified for transmission and has highest transmit priority wins the transmit acceptance filtering, and will be transmitted first. All other message objects lose the current transmit acceptance filtering round. They get a new chance in subsequent acceptance filtering rounds.

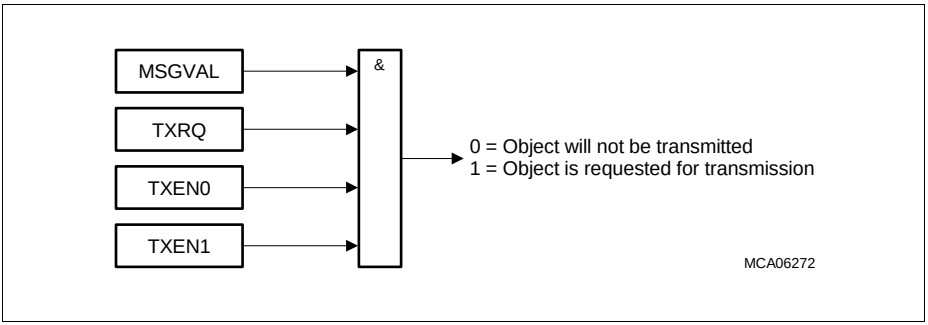


Figure 21-19 Effective Transmit Request of Message Object

21.4.10 Message Postprocessing

After a message object has successfully received or transmitted a frame, the CPU can be notified to perform a postprocessing on the message object. The postprocessing of the MultiCAN+ module consists of two elements:

1. Message interrupts to trigger postprocessing.
2. Message pending registers to collect pending message interrupts into a common structure for postprocessing.

21.4.10.1 Message Object Interrupts

When the storage of a received frame into a message object or the successful transmission of a frame is completed, a message interrupt can be issued. For each message object, a transmit and a receive interrupt can be generated and routed to one of the sixteen CAN interrupt output lines (see **Figure 21-20**). A receive interrupt occurs also after a frame storage event that has been induced by a FIFO or a gateway action. The status bits TXPND and RXPND in the Message Object n Status Register are always set after a successful transmission/reception, whether or not the respective message interrupt is enabled.

A third FIFO full interrupt condition of a message object is provided. If bit field MOFCRn.OVIE (Overflow Interrupt Enable) is set, the FIFO full interrupt will be activated depending on the actual message object type.

In case of a Receive FIFO Base Object (MOFCRn.MMC = 0001_B), the FIFO full interrupt is routed to the interrupt output line INT_Om as defined by the transmit interrupt node pointer MOIPRn.TXINP.

In case of a Transmit FIFO Base Object (MOFCRn.MMC = 0010_B), the FIFO full interrupt becomes routed to the interrupt output line INT_Om as defined by the receive interrupt node pointer MOIPRn.RXINP.

See also **“Interrupt Control” on Page 21-174** for further processing of the message object interrupts.

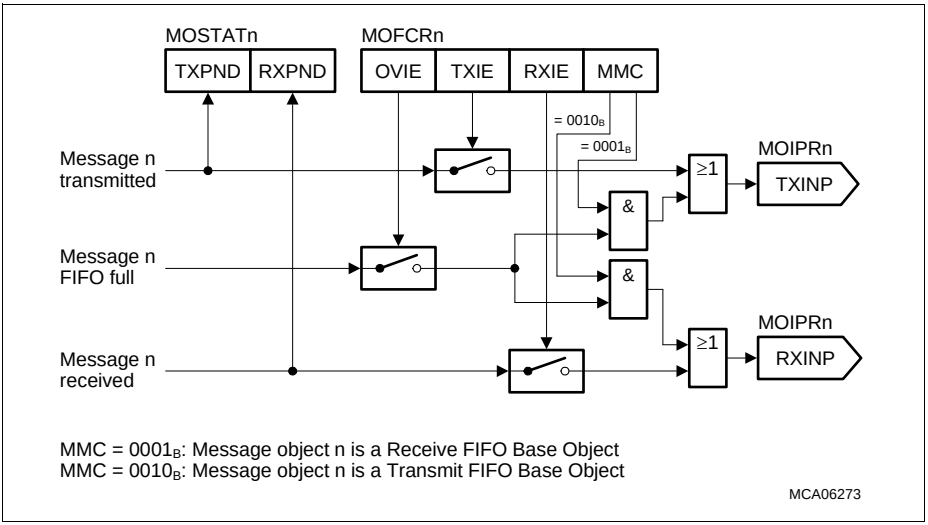


Figure 21-20 Message Interrupt Request Routing

21.4.10.2 Pending Messages

When a message interrupt request is generated, a message pending bit is set in one of the Message Pending Registers. There are 8 Message Pending Registers, MSPNDk (k = 0-7) with 32 pending bits available each. The general **Figure 21-21** shows the allocation of the message pending bits in case that the maximum possible number of eight Message Pending Registers are implemented and available on the chip.

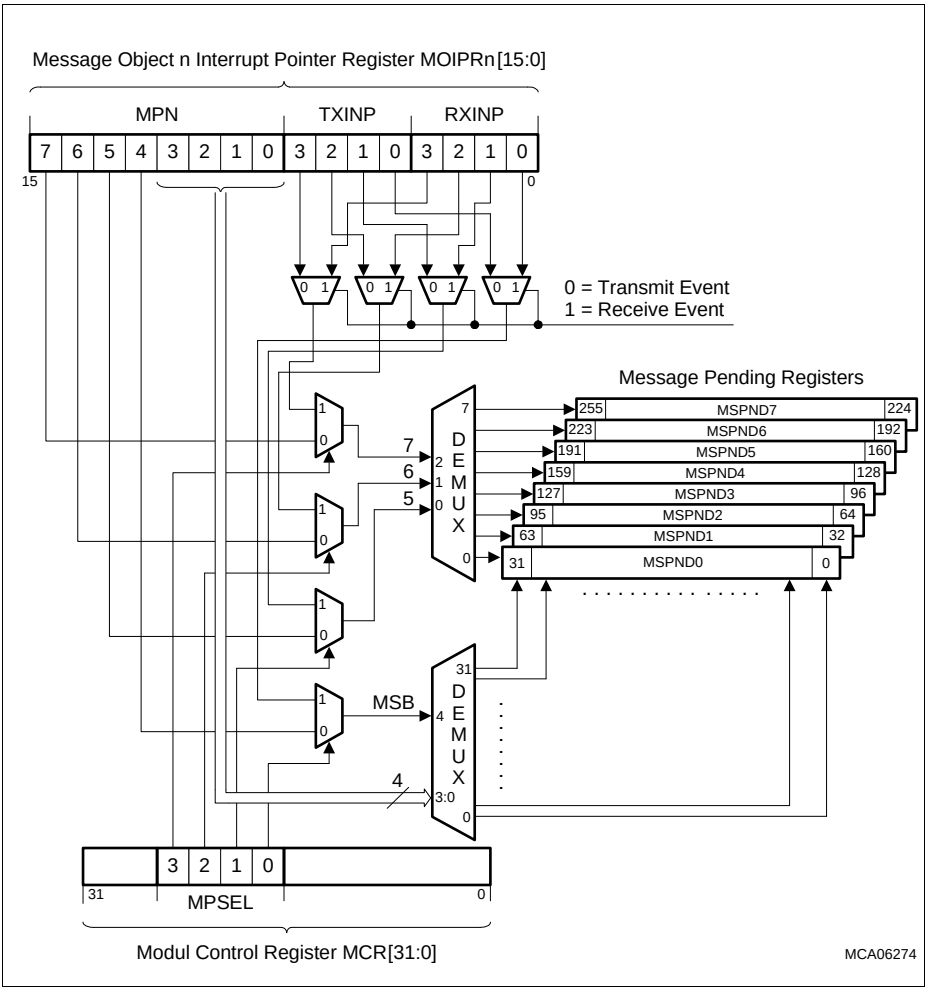


Figure 21-21 Message Pending Bit Allocation

The location of a pending bit is defined by two demultiplexers selecting the number k of the MSPNDk registers (3-bit demux), and the bit location within the corresponding MSPNDk register (5-bit demux).

Allocation Case 1

In this allocation case, bit field MCR.MPSEL = 0000_B (see **Page 21-84**). The location selection consists of 2 parts:

- The upper three bits of MOIPRn.MPN (MPN[7:5]) select the number k of a Message Pending Register MSPNDk in which the pending bit will be set.
- The lower five bits of MOIPRn.MPN (MPN[4:0]) select the bit position (0-31) in MSPNDk for the pending bit to be set.

Allocation Case 2

In this allocation case, bit field MCR.MPSEL is taken into account for pending bit allocation. Bit field MCR.MPSEL makes it possible to include the interrupt request node pointer for reception (MOIPRn.RXINP) or transmission (MOIPRn.TXINP) for pending bit allocation in such a way that different target locations for the pending bits are used in receive and transmit case. If MPSEL = 1111_B, the location selection operates in the following way:

- At a transmit event, the upper 3 bits of TXINP determine the number k of a Message Pending Register MSPNDk in which the pending bit will be set. At a receive event, the upper 3 bits of RXINP determine the number k.
- The bit position (0-31) in MSPNDk for the pending bit to be set is selected by the lowest bit of TXINP or RXINP (selects between low and high half-word of MSPNDk) and the four least significant bits of MPN.

General Hints

The Message Pending Registers MSPNDk can be written by software. Bits that are written with 1 are left unchanged, and bits which are written with 0 are cleared. This makes it possible to clear individual MSPNDk bits with a single register write access. Therefore, access conflicts are avoided when the MultiCAN+ module (hardware) sets another pending bit at the same time when software writes to the register.

Each Message Pending Register MSPNDk is associated with a Message Index Register MSIDk (see **Page 21-90**) which indicates the lowest bit position of all set (1) bits in Message Pending Register k. The MSIDk register is a read-only register that is updated immediately when a value in the corresponding Message Pending Register k is changed by software or hardware.

21.4.11 Message Object Data Handling

This chapter describes the handling capabilities for the Message Object Data of the MultiCAN+ module.

21.4.11.1 Frame Reception

After the reception of a message, it is stored in a message object according to the scheme shown in **Figure 21-22**. The MultiCAN+ module not only copies the received data into the message object, and it provides advanced features to enable consistent data exchange between MultiCAN+ and CPU.

MSGVAL

Bit MSGVAL (Message Valid) in the Message Object n Status Register MOSTATn is the main switch of the message object. During the frame reception, information is stored in the message object only when MSGVAL = 1. If bit MSGVAL is reset by the CPU, the MultiCAN+ module stops all ongoing write accesses to the message object. Now the message object can be re-configured by the CPU with subsequent write accesses to it without being disturbed by the MultiCAN+.

RTSEL

When the CPU re-configures a message object during CAN operation (for example, clears MSGVAL, modifies the message object and sets MSGVAL again), the following scenario can occur:

1. The message object wins receive acceptance filtering.
2. The CPU clears MSGVAL to re-configure the message object.
3. The CPU sets MSGVAL again after re-configuration.
4. The end of the received frame is reached. As MSGVAL is set, the received data is stored in the message object, a message interrupt request is generated, gateway and FIFO actions are processed, etc.

After the re-configuration of the message object (after step 3 above) the storage of further received data may be undesirable. This can be achieved through bit MOSTATn.RTSEL (Receive/Transmit Selected) that makes it possible to disconnect a message object from an ongoing frame reception.

When a message object wins the receive acceptance filtering, its RTSEL bit is set by the MultiCAN+ module to indicate an upcoming frame delivery. The MultiCAN+ module checks RTSEL whether it is set on successful frame reception to verify that the object is still ready for receiving the frame. The received frame is then stored in the message object (along with all subsequent actions such as message interrupts, FIFO & gateway actions, flag updates) only if RTSEL = 1.

When a message object is invalidated during CAN operation (resetting bit MSGVAL), RTSEL should be cleared before setting MSGVAL again (latest with the same write

access that sets MSGVAL) to prevent the storage of a frame that belongs to the old context of the message object. Therefore, a message object re-configuration should consist of the following steps:

1. Clear MSGVAL bit
2. Re-configure the message object while MSGVAL = 0
3. Clear RTSEL bit and set MSGVAL again

RXEN

Bit MOSTATn.RXEN enables a message object for frame reception. A message object can receive CAN messages from the CAN bus only if RXEN = 1. The MultiCAN+ module evaluates RXEN only during receive acceptance filtering. After receive acceptance filtering, RXEN is ignored and has no further influence on the actual storage of a received message in a message object.

Bit RXEN enables the “soft phase out” of a message object: after clearing RXEN, a currently received CAN message for which the message object has won acceptance filtering is still stored in the message object but for subsequent messages the message object no longer wins receive acceptance filtering.

RXUPD, NEWDAT and MSGLST

An ongoing frame storage process is indicated by the RXUPD (Receive Updating) flag in the MOSTATn register. RXUPD is set with the start and cleared with the end of a message object update, which consists of frame storage as well as flag updates.

After storing the received frame (identifier, IDE bit, DLC; including the Data Field for Data Frames), the NEWDAT (New Data) bit of the message object is set. If NEWDAT was already set before it becomes set again, bit MSGLST (Message Lost) is set to indicate a data loss condition.

The RXUPD and NEWDAT flags can help to read consistent frame data from the message object during an ongoing CAN operation. The following steps are recommended to be executed:

1. Clear NEWDAT bit.
2. Read message content (identifier, data etc.) from the message object.
3. Check that both, NEWDAT and RXUPD, are cleared. If this is not the case, go back to step 1.
4. When step 3 was successful, the message object contents are consistent and has not been updated by the MultiCAN+ module while reading.

Bits RXUPD, NEWDAT and MSGLST have the same behavior for the reception of Data as well as Remote Frames.

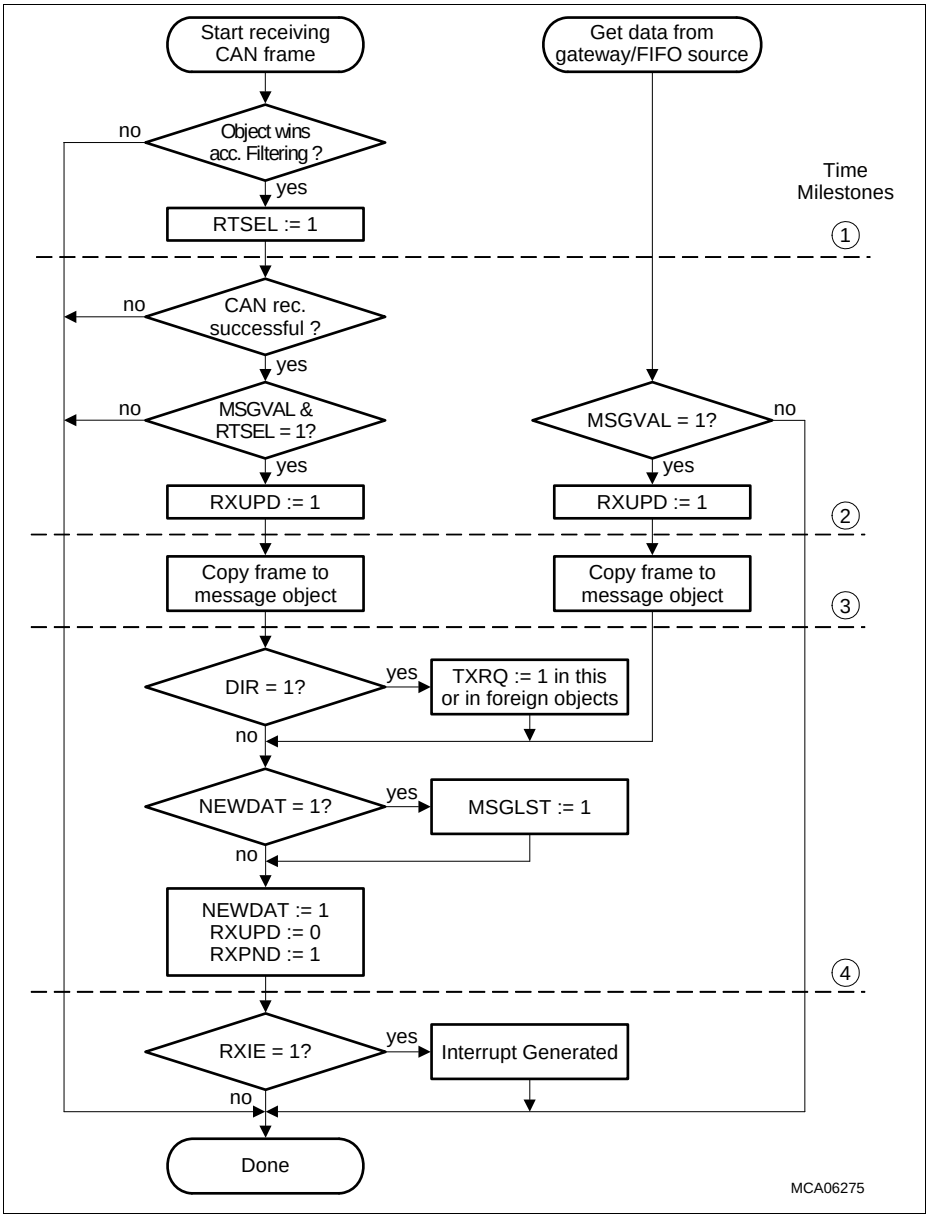


Figure 21-22 Reception of a Message Object

21.4.11.2 Frame Transmission

The process of a message object transmission is shown in **Figure 21-23**. Along with the copy of the message object content to be transmitted (identifier, IDE bit, RTR = DIR bit, DLC, including the Data Field for Data Frames) into the internal transmit buffer of the assigned CAN node, several status flags are also served and monitored to control consistent data handling.

The transmission process of a message object starting after the transmit acceptance filtering is identical for Remote and Data Frames.

MSGVAL, TXRQ, TXEN0, TXEN1

A message can only be transmitted if all four bits in registers MOSTATn, MSGVAL (Message Valid), TXRQ (Transmit Request), TXEN0 (Transmit Enable 0), TXEN1 (Transmit Enable 1) are set as shown in **Figure 21-19**. Although these bits are equivalent with respect to the transmission process, they have different semantics:

Table 21-8 Message Transmission Bit Definitions

Bit	Description
MSGVAL	**Message Valid** This is the main switch bit of the message object.
TXRQ	**Transmit Request** This is the standard transmit request bit. This bit must be set whenever a message object should be transmitted. TXRQ is cleared by hardware at the end of a successful transmission, except when there is new data (indicated by NEWDAT = 1) to be transmitted. When bit MOFCRn.STT (“Single Transmit Trial”) is set, TXRQ becomes already cleared when the contents of the message object are copied into the transmit frame buffer of the CAN node. A received remote request (after a Remote Frame reception) sets bit TXRQ to request the transmission of the requested data frame.
TXEN0	**Transmit Enable 0** This bit can be temporarily cleared by software to suppress the transmission of this message object when it writes new content to the Data Field. This avoids transmission of inconsistent frames that consist of a mixture of old and new data. Remote requests are still accepted when TXEN0 = 0, but transmission of the Data Frame is suspended until transmission is re-enabled by software (setting TXEN0).

Table 21-8 Message Transmission Bit Definitions (cont'd)

Bit	Description
TXEN1	**Transmit Enable 1** This bit is used in transmit FIFOs to select the message object that is transmit active within the FIFO structure. For message objects that are not transmit FIFO elements, TXEN1 can either be set permanently to 1 or can be used as a second independent transmission enable bit.

RTSEL

When a message object has been identified to be transmitted next after transmission acceptance filtering, bit MOSTATn.RTSEL (Receive/Transmit Selected) is set.

When the message object is copied into the internal transmit buffer, bit RTSEL is checked, and the message is transmitted only if RTSEL = 1. After the successful transmission of the message, bit RTSEL is checked again and the message postprocessing is only executed if RTSEL = 1.

For a complete re-configuration of a valid message object, the following steps should be executed:

1. Clear MSGVAL bit
2. Re-configure the message object while MSGVAL = 0
3. Clear RTSEL and set MSGVAL

Clearing of RTSEL ensures that the message object is disconnected from an ongoing/scheduled transmission and no message object processing (copying message to transmit buffer including clearing NEWDAT, clearing TXRQ, time stamp update, message interrupt, etc.) within the old context of the object can occur after the message object becomes valid again, but within a new context.

NEWDAT

When the contents of a message object have been transferred to the internal transmit buffer of the CAN node, bit MOSTATn.NEWDAT (New Data) is cleared by hardware to indicate that the transmit message object data is no longer new.

When the transmission of the frame is successful and NEWDAT is still cleared (if no new data has been copied into the message object meanwhile), TXRQ (Transmit Request) is cleared automatically by hardware.

If, however, the NEWDAT bit has been set again by the software (because a new frame should be transmitted), TXRQ is not cleared to enable the transmission of the new data.

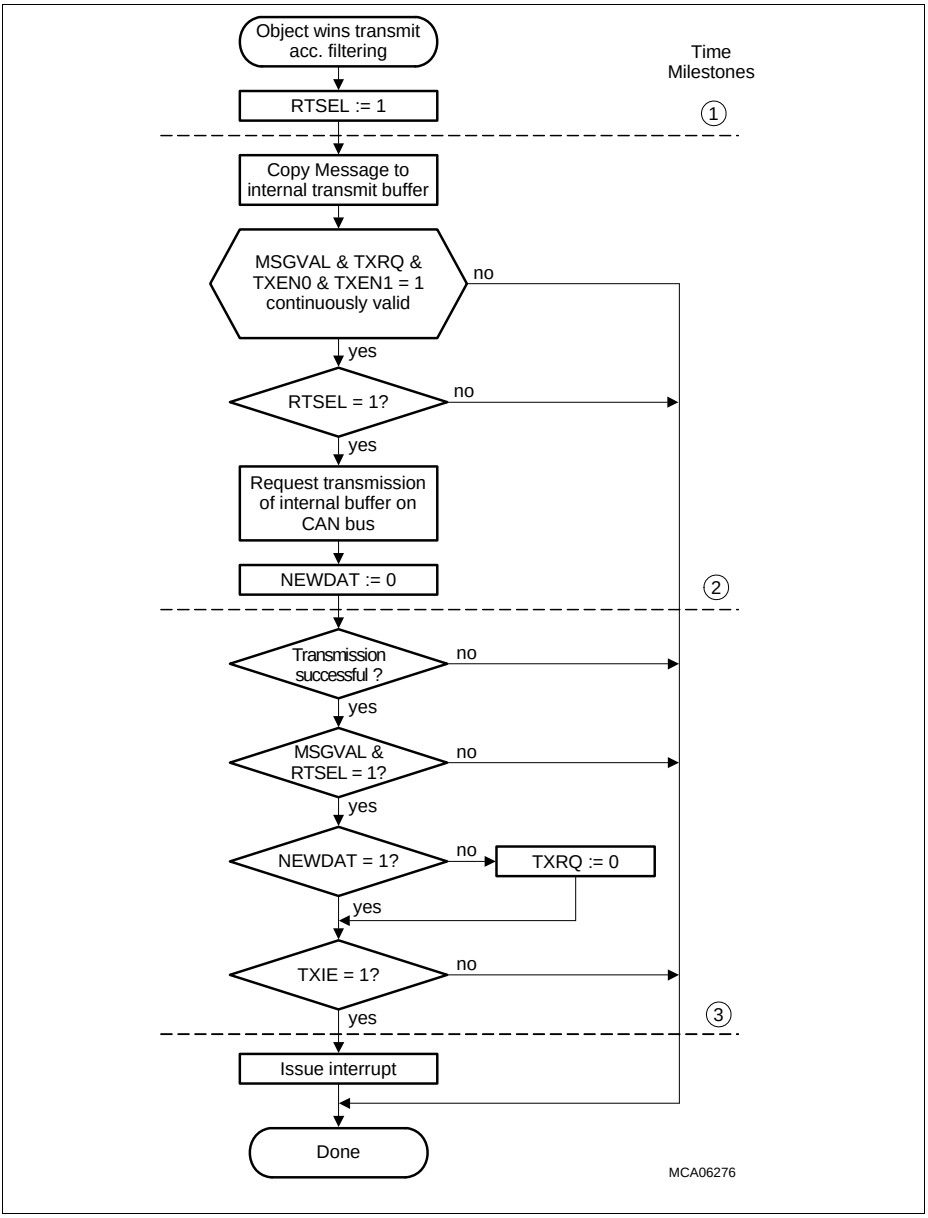


Figure 21-23 Transmission of a Message Object

21.4.12 Message Object Functionality

This chapter describes the functionality of the Message Objects in the MultiCAN+ module.

21.4.12.1 Standard Message Object

A message object is selected as standard message object when bit field MOFCRn.MMC = 0000_B (see **Page 21-114**). The standard message object can transmit and receive CAN frames according to the basic rules described in the previous sections. Additional services such as Single Data Transfer Mode or Single Transmit Trial (see following sections) are available and can be individually selected.

21.4.12.2 Single Data Transfer Mode

Single Data Transfer Mode is a useful feature in order to broadcast data over the CAN bus without unintended duplication of information. Single Data Transfer Mode is selected via bit MOFCRn.SDT.

Message Reception

When a received message stored in a message object is overwritten by a new received message, the contents of the first message are lost and replaced with the contents of the new received message (indicated by MSGLST = 1).

If SDT is set (Single Data Transfer Mode activated), bit MSGVAL of the message object is automatically cleared by hardware after the storage of a received Data or Remote Frame. This prevents the reception of further messages.

Message Transmission

When a message object receives a series of multiple remote requests, it transmits several Data Frames in response to the remote requests. If the data within the message object has not been updated in the time between the transmissions, the same data can be sent more than once on the CAN bus.

In Single Data Transfer Mode (SDT = 1), this is avoided because MSGVAL is automatically cleared after the successful transmission of a Data or Remote Frame.

21.4.12.3 Single Transmit Trial

If the bit STT in the message object function register is set (STT = 1), the transmission request is cleared (TXRQ = 0) when the frame contents of the message object have been copied to the internal transmit buffer of the CAN node. Thus, the transmission of the message object is not tried again when it fails due to CAN bus errors.

21.4.12.4 Message Object Format (Classical CAN & CAN FD)

Message objects are used for transmission or reception of CAN data frames. The transmit and receive behavior of a Classical CAN node in Classical CAN (ISO 11898-1:2003(E)) or CAN FD (ISO11898-1 DIS version 2014) are determined by a node control register bit (i.e NCRx.FDEN) and two message object function control register bits (i.e MOFCRn.FDF and MOFCRn.BRS) as shown in **Table 21-18**.

A CAN node which has been set to work in Classical CAN mode i.e (NCRx.FDEN = 0) would be able to transmit message objects programmed for Classical CAN Frames (i.e MOFCRn.FDF = 0, MOFCRn.BRS = 0/1) only. When receiving Classical CAN data frames in a node that works in Classical CAN mode the message object function control register bits (i.e MOFCRn.FDF and MODFCRn.BRS) are not used and not updated.

In the case when a CAN node set to work in a particular mode (e.g Classical CAN mode) and receives a message object request for transmission or reception that does not correspond with the CAN mode (i.e MOFCRn.FDF = 1, MOFCRn.BRS = 0/1), the transmission would be cancelled (i.e corresponding MOSTATn.TXRQ cleared) and CAN frame not received in message object. The CAN node is not blocked and able to transmit or receive other message objects.

However a CAN node that has been set to work in CAN FD mode i.e (NCRx.FDEN = 1) is able to transmit and receive Classical CAN Data Frames, Long Data Frames or Long + Fast Data Frames. For message object transmission, message object function control bits (i.e MOFCRn.FDF & MOFCRn.BRS) determines the CAN frame format to be used and functions as status register bits during message reception.

Note: Remote Frame Requests do not change MOFCRn.FDF and MOFCRn.BRS bits.

21.4.12.5 Message Object FIFO Structure

In case of high CPU load it may be difficult to process a series of CAN frames in time. This may happen if multiple messages are received or must be transmitted in short time.

Therefore, a FIFO buffer structure is available to avoid loss of incoming messages and to minimize the setup time for outgoing messages. The FIFO structure can also be used to automate the reception or transmission of a series of CAN messages and to generate a single message interrupt when the whole CAN frame series is done.

There can be several FIFOs in parallel. The number of FIFOs and their size are limited only by the number of available message objects. A FIFO can be installed, resized and de-installed at any time, even during CAN operation.

The basic structure of a FIFO is shown in **Figure 21-24**. A FIFO consists of one base object and n slave objects. The slave objects are chained together in a list structure (similar as in message object lists). The base object may be allocated to any list. Although **Figure 21-24** shows the base object as a separate part beside the slave objects, it is also possible to integrate the base object at any place into the chain of slave objects. This means that the base object is slave object, too (not possible for gateways).

The absolute object numbers of the message objects have no impact on the operation of the FIFO.

The base object does not need to be allocated to the same list as the slave objects. Only the slave object must be allocated to a common list (as they are chained together). Several pointers (BOT, CUR and TOP) that are located in the Message Object n FIFO/Gateway Pointer Register MOFGPRn link the base object to the slave objects, regardless whether the base object is allocated to the same or to another **list** than the slave objects.

The smallest FIFO would be a single message object which is both, FIFO base and FIFO slave (not very useful). The biggest possible FIFO structure would include all message objects of the MultiCAN+ module. Any FIFO sizes between these limits are possible.

In the FIFO base object, the FIFO boundaries are defined. Bit field MOFGPRn.BOT of the base object points to (includes the number of) the bottom slave object in the FIFO structure. The MOFGPRn.TOP bit field points to (includes the number of) the top slave object in the FIFO structure. The MOFGPRn.CUR bit field points to (includes the number of) the slave object that is actually selected by the MultiCAN+ module for message transfer. When a message transfer takes place with this object, CUR is set to the next message object in the list structure of the slave objects (CUR = PNEXT of current object). If CUR was equal to TOP (top of the FIFO reached), the next update of CUR will result in CUR = BOT (wrap-around from the top to the bottom of the FIFO). This scheme represents a circular FIFO structure where the bit fields BOT and TOP establish the link from the last to the first element.

Bit field MOFGPRn.SEL of the base object can be used for monitoring purposes. It makes it possible to define a slave object within the list at which a message interrupt is generated whenever the CUR pointer reaches the value of the SEL pointer. Thus SEL makes it possible to detect the end of a predefined message transfer series or to issue a warning interrupt when the FIFO becomes full.

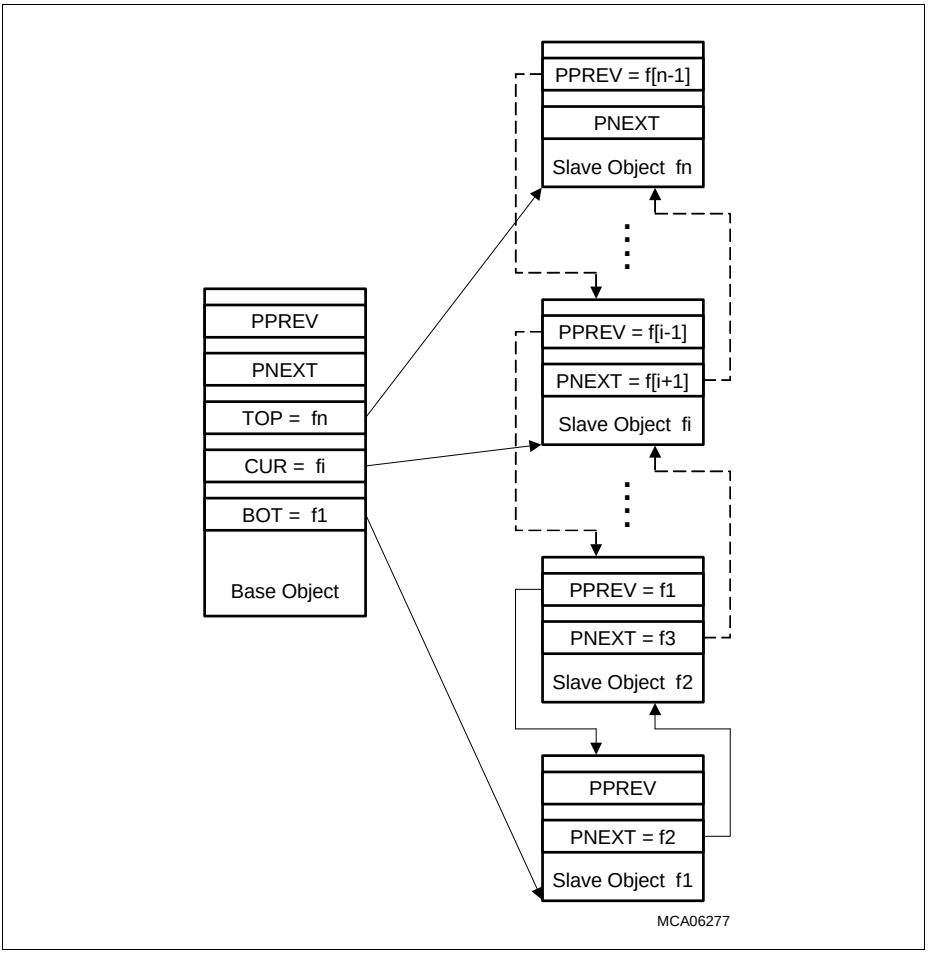


Figure 21-24 FIFO Structure with FIFO Base Object and n FIFO Slave Objects

21.4.12.6 Receive FIFO

The Receive FIFO structure is used to buffer incoming (received) Remote or Data Frames.

A Receive FIFO is selected by setting MOFCRn.MMC = 0001_B in the FIFO base object. This MMC code automatically designates a message object as FIFO base object. The message modes of the FIFO slave objects are not relevant for the operation of the Receive FIFO.

When the FIFO base object receives a frame from the CAN node it belongs to, the frame is not stored in the base object itself but in the message object that is selected by the base object's MOFGPRn.CUR pointer. This message object receives the CAN message as if it is the direct receiver of the message. However, MOFCRn.MMC = 0000_B is implicitly assumed for the FIFO slave object, and a standard message delivery is performed. The actual message mode (MMC setting) of the FIFO slave object is ignored. For the slave object, no acceptance filtering takes place that checks the received frame for a match with the identifier, IDE bit, and DIR bit.

With the reception of a CAN frame, the current pointer CUR of the base object is set to the number of the next message object in the FIFO structure. This message object will then be used to store the next incoming message.

If bit field MOFCRn.OVIE ("Overflow Interrupt Enable") of the FIFO base object is set and the current pointer MOFGPRn.CUR becomes equal to MOFGPRn.SEL, a FIFO overflow interrupt request is generated. This interrupt request is generated on interrupt node TXINP of the base object immediately after the storage of the received frame in the slave object. Transmit interrupts are still generated if TXIE is set.

A CAN message is stored in FIFO base and slave object only if MSGVAL = 1.

In order to avoid direct reception of a message by a slave message object, as if it was an independent message object and not a part of a FIFO, the bit RXEN of each slave object must be cleared. The setting of the bit RXEN is "don't care" only if the slave object is located in a list not assigned to a CAN node.

21.4.12.7 Transmit FIFO

The Transmit FIFO structure is used to buffer a series of Data or Remote Frames that must be transmitted. A transmit FIFO consists of one base message object and one or more slave message objects.

A Transmit FIFO is selected by setting MOFCRn.MMC = 0010_B in the FIFO base object. Unlike the Receive FIFO, slave objects assigned to the Transmit FIFO must explicitly set their bit fields MOFCRn.MMC = 0011_B. The CUR pointer in all slave objects must point back to the Transmit FIFO Base Object (to be initialized by software).

The MOSTATn.TXEN1 bits (Transmit Enable 1) of all message objects except the one which is selected by the CUR pointer of the base object must be cleared by software. TXEN1 of the message (slave) object selected by CUR must be set. CUR (of the base object) may be initialized to any FIFO slave object.

When tagging the message objects of the FIFO as valid to start the operation of the FIFO, then the base object must be tagged valid (MSGVAL = 1) first.

Before a Transmit FIFO becomes de-installed during operation, its slave objects must be tagged invalid (MSGVAL = 0).

The Transmit FIFO uses the TXEN1 bit in the Message Object Status Register of all FIFO elements to select the actual message for transmission. Transmit acceptance filtering evaluates TXEN1 for each message object and a message object can win transmit acceptance filtering only if its TXEN1 bit is set. When a FIFO object has transmitted a message, the hardware clears its TXEN1 bit in addition to standard transmit postprocessing (clear TXRQ, transmit interrupt etc.), and moves the CUR pointer in the next FIFO base object to be transmitted. TXEN1 is set automatically (by hardware) in the next message object. Thus, TXEN1 moves along the Transmit FIFO structure as a token that selects the active element.

If bit field MOFCRn.OVIE (“Overflow Interrupt Enable”) of the FIFO base object is set and the current pointer CUR becomes equal to MOFGPRn.SEL, a FIFO overflow interrupt request is generated. The interrupt request is generated on interrupt node RXINP of the base object after postprocessing of the received frame. Receive interrupts are still generated for the Transmit FIFO base object if bit RXIE is set.

21.4.12.8 Gateway Mode

The Gateway Mode makes it possible to establish an automatic information transfer between two independent CAN buses without CPU interaction.

The Gateway Mode operates on message object level. In Gateway mode, information is transferred between two message objects, resulting in an information transfer between the two CAN nodes to which the message objects are allocated. A gateway may be established with any pair of CAN nodes, and there can be as many gateways as there are message objects available to build the gateway structure.

Gateway Mode is selected by setting MOFCRs.MMC = 0100_B for the gateway source object s. The gateway destination object d is selected by the MOFGPRd.CUR pointer of the source object. The gateway destination object only needs to be valid (its MSGVAL = 1). All other settings are not relevant for the information transfer from the source object to the destination object.

Gateway source object behaves as a standard message object with the difference that some additional actions are performed by the MultiCAN+ module when a CAN frame has been received and stored in the source object (see **Figure 21-25**):

1. If bit MOFCRs.DLCC is set, the data length code MOFCRs.DLC is copied from the gateway source object to the gateway destination object.
2. If bit MOFCRs.IDC is set, the identifier MOARs.ID and the identifier extension MOARs.IDE are copied from the gateway source object to the gateway destination object.
3. If bit MOFCRs.DATC is set, the data bytes stored in the two data registers MODATALs and MODATAHs are copied from the gateway source object to the gateway destination object. All 8 data bytes are copied, even if MOFCRs.DLC indicates less than 8 data bytes.
4. If bit MOFCRs.GDFS is set, the transmit request flag MOSTATd.TXRQ is set in the gateway destination object.
5. The receive pending bit MOSTATd.RXPND and the new data bit MOSTATd.NEWDAT are set in the gateway destination object.
6. A message interrupt request is generated for the gateway destination object if its MOSTATd.RXIE is set.
7. The current object pointer MOFGPRs.CUR of the gateway source object is moved to the next destination object according to the FIFO rules as described on **Page 21-57**. A gateway with a single (static) destination object is obtained by setting MOFGPRs.TOP = MOFGPRs.BOT = MOFGPRs.CUR = destination object.

The link from the gateway source object to the gateway destination object works in the same way as the link from a FIFO base to a FIFO slave. This means that a gateway with an integrated destination FIFO may be created; in **Figure 21-24**, the object on the left is the gateway source object and the message object on the right side is the gateway destination objects.

The gateway operates equivalent for the reception of data frames (source object is receive object, i.e. DIR = 0) as well as for the reception of Remote Frames (source object is transmit object).

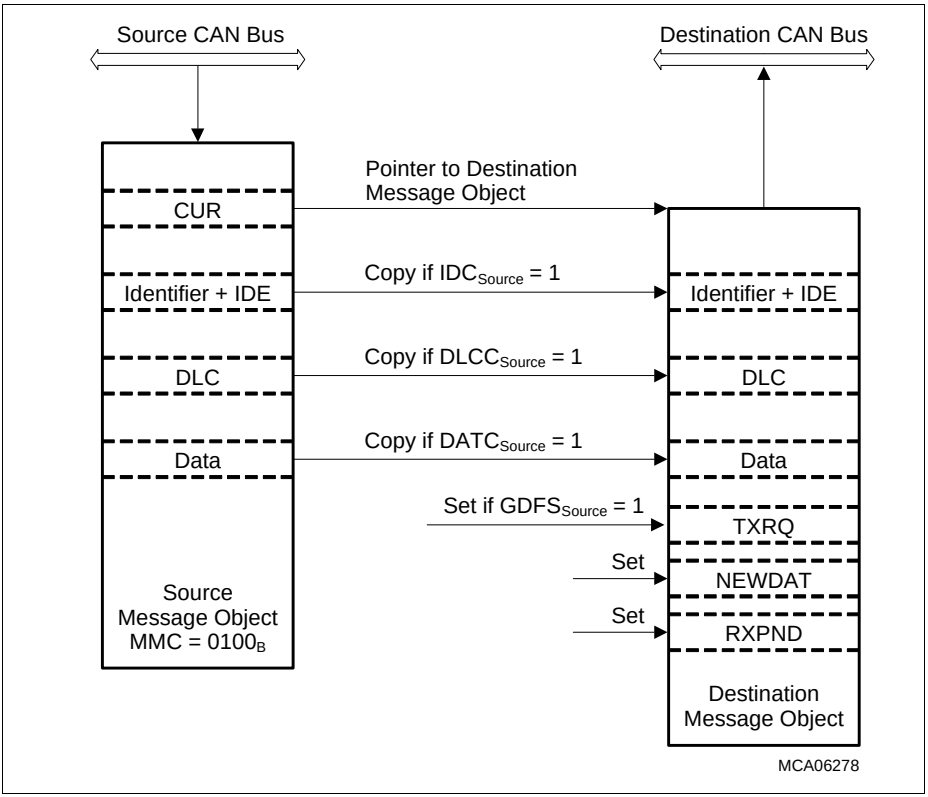


Figure 21-25 Gateway Transfer from Source to Destination

21.4.12.9 Foreign Remote Requests

When a Remote Frame has been received on a CAN node and is stored in a message object, a transmit request is set to trigger the answer (transmission of a Data Frame) to the request or to automatically issue a secondary request. If the Foreign Remote Request Enable bit MOFCRn.FRREN is cleared in the message object in which the remote request is stored, MOSTATn.TXRQ is set in the same message object.

If bit FRREN is set (FRREN = 1: foreign remote request enabled), TXRQ is set in the message object that is referenced by pointer MOFGPRn.CUR. The value of CUR is, however, not changed by this feature.

Although the foreign remote request feature works independently of the selected message mode, it is especially useful for gateways to issue a remote request on the source bus of a gateway after the reception of a remote request on the gateway destination bus. According to the setting of FRREN in the gateway destination object, there are two capabilities to handle remote requests that appear on the destination side (assuming that the source object is a receive object and the destination is a transmit object, i.e. DIR_{source} = 0 and $DIR_{destination}$ = 1):

FRREN = 0 in the Gateway Destination Object

1. A Remote Frame is received by gateway destination object.
2. TXRQ is set automatically in the gateway destination object.
3. A Data Frame with the current data stored in the destination object is transmitted on the destination bus.

FRREN = 1 in the Gateway Destination Object

1. A Remote Frame is received by gateway destination object.
2. TXRQ is set automatically in the gateway source object (must be referenced by CUR pointer of the destination object).
3. A remote request is transmitted by the source object (which is a receive object) on the source CAN bus.
4. The receiver of the remote request responds with a Data Frame on the source bus.
5. The Data Frame is stored in the source object.
6. The Data Frame is copied to the destination object (gateway action).
7. TXRQ is set in the destination object (assuming $GDFS_{source}$ = 1).
8. The new data stored in the destination object is transmitted on the destination bus, in response to the initial remote request on the destination bus.

21.4.12.10CAN FD - 64 byte Messages

In order to support higher than 8 data bytes payload (e.g 64bytes data payload) using current message object structure the data byte buffer (MODATALn and MODATAHn) has to be extended.

Thus an additional CAN FD 64 bytes message mode, MOFCR.MMC=5 is added. When CAN FD 64 bytes message mode is selected, additional message objects are used to store the extra data bytes.

The additional message objects used are specified by the pointer on MOFGPR.BOT and MOFGPR.TOP register bits. (i.e those message object registers (MOFCR, MOFGPR, MOIPR, MOAMR, MODATAL, MODATAH and MOAR) are used to store the extra data bytes with their alternate register view on EMOzDATAn at the same address location. MOCTR is the only register that is not used for data storage it retains its original function.

As an example for a 64 byte message, data bytes 0-7 are stored in the message object, similar to a standard message object, data bytes 8-35 are stored in the message object to which MOFGPR.BOT refers to and data bytes 36-63 are stored in the message object to which MOFGPR.TOP refers to. Data byte 0 refers to the first byte transferred within a CAN frame. For shorter than 64 bytes messages, unused data bytes are padded upon reception and ignored upon transmission.

The additional message objects chosen to store the extra data bytes must not take part in CAN communication i.e either they are allocated to a list that does not belong to an active CAN node or have that particular message object MOCTR register bits MOCTRn.RXEN, MOCTRn.TXEN0 (and/or TXEN1, TXRQ) cleared, i.e (RXEN=0, TXEN0=0, TXEN1=0 and TXRQ=0). MOCTR.MSGVAL remains the same, data is stored in the extra message objects when MSGVAL is set. When MSGVAL of the base object is cleared, an ongoing data transfer to TOP and BOT objects are aborted, thereby freeing objects BOT and TOP. Clearing MSGVAL bits of BOT, TOP objects and the extra message objects do not stop an ongoing data transfer and are not considered at all.

For optimum performance it is recommended to place the extra message objects outside active CAN node lists, so they do not take part in CAN acceptance filtering.

21.4.13 Measurement for Oscillator Calibration

The chip can be driven by an oscillator which is less accurate than required by the CAN specification, if it is possible to detect the deviation and to change the oscillator frequency during runtime. The MultiCAN module can measure the time interval between the start of a CAN frame and a certain edge on the bus. The timer hardware is accessed via Measurement Control Register MECR (see **Page 21-92**) and Measurement Status Register MESTAT (see **Page 21-94**).

A 16-bit timer is cleared and started by the first falling (dominant) edge of a CAN frame on the input line of the CAN node (selected by MECR.NODE). The timer is incremented by the module control clock f_{CLC}. The timer content is captured when an edge specified by MECR.ANYED is detected on the input line of the CAN node. However, the timer must have reached threshold MECR.TH before; so edges occurring earlier cannot cause a capture. Only the first edge after threshold is considered, and the node itself may not be sending. Bit field CAPT in the status register contains the captured timer value, and bit CAPRED indicates the type of edge that caused the capture. On a capture event, bit

CAPE in the same register will be set. If bit MECR.CAPEIE = 1, a node interrupt will be generated. With the starting edge of the following CAN frame, a new measurement cycle begins. To avoid an overflow on the measurement and to get a measurement result MESTAT.CAPE bit has to be cleared before the measurement end condition occurs.

To avoid wrong capture results due to timer overflow, the timer will be stopped when $FFFF_H$ is reached. If MECR.TH = 0000_H, the timer is always stopped and the capture function is disabled.

Some important issues should be mentioned:

- The timing of the measured CAN frame should be as exact as possible.
- No unadjusted CAN stations - including the selected MultiCAN node - should participate actively in the communication. The selected node can be set to Analyzer Mode.
- When setting the timer threshold and evaluating the capture result, the tolerances including the oscillator deviation must be considered.
- It may not be possible to use a well-defined CAN message for measurement. Then worst-case-scenarios according to the CAN protocol must be observed. E.g. in a data or remote frame, the first rising (recessive) edge might occur after five bit times, and the next falling one after additional five bit times.
- Also error and overload frames which may lead to unexpected capture results should be taken into account.
- Edges occurring a considerable time after the start of a CAN frame will yield good resolution but may produce ambiguous results if the oscillator is not yet sufficiently adjusted. Therefore, only one step for measurement and calibration may not be sufficient.
- As there is only one oscillator on the chip, its calibration influences the timing of the whole system and particularly the timing of other CAN nodes.

21.4.14 Debug Over CAN

The MultiCAN+ controller supports debugging[1)] using CAN tool access in parallel to regular CAN bus traffic. This is achieved by transmitting DAP telegrams and replies as regular CAN messages (DXCM DAP over CAN Messages). DXCM uses the two highest message objects and it is strongly recommended to use also the same CAN pins as for DXCPL (DAP over CAN Physical Layer). DXCM is enabled with the MCR.DXCM bit. Please refer to the OCDS chapter for more information about DAP, DXCM and DXCPL.

1) Debug over CAN feature is available through CAN module only and not available on CAN1 module.

21.5 Use Case Example MultiCAN+

This section explains the core functionality of the MultiCAN+ module with a code example. The example realizes sending a CAN message via internal CAN-bus from node 0 to node 1.

The MultiCAN+ module for the TC21x/TC22x/TC23x consists of 2[1)] modules (i.e MultiCAN with 3 CAN nodes and MultiCAN1 with 3 CAN Nodes), representing 6[2)] serial communication interfaces. Each CAN node can either be connected to a port or to the internal CAN bus (see). So a maximum of 6[2)] CAN channels can be realized with the MultiCAN+ module (For more information and further functions see **Section 21.2**).

All CAN nodes share a common set of 128/128 message objects. A message object function like a container for a specific CAN message; both transmitting and receiving of CAN messages can be realized. The message objects are organized in double-chained lists, each list is dedicated to a certain CAN node (list 1 is node 0, list 2 is node 1, etc.). After a reset, all message objects are unallocated and therefore assigned to list 0, this list does not belong to a CAN node. During initialization it is possible to allocate the message objects individually to certain CAN node lists. Each allocated message object can be set up with all necessary information like the message ID etc. (see chapter 26.2 and following for further explanations of the CAN module).

In the use case example CAN node 0 will send a CAN message via internal CAN bus to CAN node 1. The transmitting message object (MO) will be MO 0 and the receiving message object will be MO 1. As soon as the CAN frame successfully receives at MO 1 a receive interrupt will occur. The initialization process follows the following order:

1. Load global MuliCAN+ module registers
2. Initialize the CAN nodes
3. Allocate the message objects to the CAN nodes
4. Initialize the message objects
5. Start the CAN nodes
6. Start transmit request for CAN message from node 0 to node 1.
7. Receive message at node 1, Rx interrupt occurs.

Step description to initialize the MultiCAN+ module:

(Line 1) reset ENDINIT to get access to ENDINIT protected registers. (see also ENDINIT protection chapter 8.9.3(LINK)).

(Line 2) enable control of the module, in clock control register CLC (**CLC**).

(Line 3) read back (dummy variable has to be defined). The reading process ensures that the write process from line 2 is done.

1) TC23x consists of 2 modules (MultiCAN and MultiCAN1), TC22x consists of 1 module (MultiCAN).

2) TC23x has 6 CAN serial communication interfaces, TC22x has 3 CAN communication interfaces.

(Line 4) load fractional divider to $f_{CAN} = f_{ASYN_CAN}$ (see also 26.3.2 Clock Control (**Fractional Divider**), f_{ASYN_CAN} see **MCR**).

(Line 5) set ENDINIT to lock the protected register again.

(Line 6) set clk source to f_{ASYN_CAN} in module control register MCR(**MCR**).

Note: The reconfiguration of the clk source must be done by two writes and a certain delay between these. See MCR for more details.

(Line 7) The setting of protection CCE[6] and INIT[0] activates the initialization and configuration mode for CAN node 0. This is necessary for the upcoming changes in CAN node 0. (see also node configuration register NCRn (**CAN_NCRx (x = 0-2)**))

(Line 8) This example uses the internal loop-back mode, so this line connects CAN node 0 to the internal CAN Bus. (see also Figure 26-13(**Figure 21-17**) and node port control register NPCR(**CAN_NPCRx (x = 0-2)**))

(Line 9) The CAN bit time is subdivided into different segments. (see also 26.3.6.1 Bit Timing Unit(**Bit Timing Unit**)). Assuming f_{CAN} = 100MHz this line then sets the baudrate to 500kBaud and the sample point to 80% in the node bit timing register NBTR (**CAN_NBTRx (x = 0-2)**).

Note: The FM PLL should be off. Otherwise a stable CAN communication is not ensured

(Line 10 - 12) same as line 7 - 9 but with node 1.

(Line 13) The allocation of the message objects to certain CAN node lists function via a list command panel. The panel control register PANCTR(**PANCTR**) can only be written if both busy flags are reseted. Therefore this while loop waits until the register is ready.

(Line 14) This line allocates message object 0 to list 1 (belongs to CAN node 0).

(Line 15) see line 13.

(Line 16) This line allocates message object 1 to list 2 (belongs to CAN node 0).

(Line 17) Initialize MO 0 in the message object register MOCTR(**CAN_MOCTRz (z = 0-126)**) as transmit MO by setting bits [27:25]. Set also MSGVAL[21], that's the main switch bit of the MO.

(Line 18) this line sets the message data length of MO 0 to 8 bytes.(see also message object function control register MOFCR (**CAN_MOFCRn (n = 0-127)**)).

(Line 19) the message ID of MO 0 gets set to standard message (11 bit length) with a message ID of 0xFF in the Message object arbitration register MOAR (**CAN_MOARn (n = 0-127)**).

(Line 20) Initialize MO 1 in the message object register MOCTR as receive MO by setting bit RXEN[23]. Set also MSGVAL[21], that's the main switch bit of the MO.

(Line 21) the receive interrupt gets enabled in the message object function control register MOFCR .

(Line 22) same as Line 19 but for MO 1. The same message ID is necessary to receive CAN messages from the bus with this specific ID.

(Line 23) the Rx interrupt gets connected to the CAN interrupt output line 1.(see also MOIPR(**CAN_MOIPRn (n = 0-127)**))

(Line 24) This line enables the Rx interrupt in the service request control register SRC_CANINT1 and sets the interrupt priority to CAN_SR1INT (1...255)

(Line 25) The interruptHandlerInstall function gets called (this function has to be written first, see interrupt handler example (Chapter 17.10.1) for more details). This function installs the interrupt service routine (ISR) entry address in the interrupt vector array with the priority CAN_SRN1INT.

(Line 26) The 4 lower data bytes are getting loaded in the MODATAL(**CAN_MODATALn (n = 0-127)**) register. the data itself is here just an example.

(Line 27) The 4 higher data bytes are getting loaded in the MODATAH(**CAN_MODATAHn (n = 0-127)**) register. (just an data example as well)

(Line 28) This line resets INIT[0] and CCE[6] bit in the node control register from node 0. The node is now synchronizing itself to the bus.

(Line 29) same as line 28, but with node 1.

(Line 30) The NEWDATA bit gets set in the MOCTR register, this should always be done after a write process in the data registers from line 26/27. The transmit request bit TXRQ gets set, this starts the transmission of the CAN message. The RTSEL gets reset to ensure the transmission.

(Line 31) If the message is received the MOCTR.NEWDAT[19] bit in the receive MO 1 is set. This line just waits for the reception. The occurrence of the Rx interrupt can also be used as a proof that the message arrived.

Note: The Rx ISR function prototype would be: void CAN1_Rx_irq (void); here could be your ISR code;

Initialization of the MultiCAN+ module:

```
// Load global MultiCAN+ registers:
(1)  SCU_vResetENDINIT (0);     // enable ENDINIT reg
(2)  CAN_CLC = 0x000;           // disable module control
(3)  dummy = CAN_CLC;           //read to check the made changes
(4)  CAN_FDR = (1 << 14)| 0x3FF;     //DM=1,STEP=1023
(5)  SCU_vSetENDINIT (0);       // lock ENDINIT reg
(6)  CAN_MCR = 0x1;             // CLKSEL=1
//Init CAN nodes 0 and 1:
(7)  CAN_NCR0 = (1 << 6)| 1;    // CCE=1,INIT=1
```

```
(8)  CAN_NPCR0 = (1 << 8);      // LBM=1
(9)  CAN_NBTR0 = 0x3EC9;        //set bit segments
(10) CAN_NCR1 = (1 << 6)| 1;    // CCE=1,INIT=1
(11) CAN_NPCR1 = (1 << 8);      // LBM=1
(12) CAN_NBTR1 = 0x3EC9;        //set bit segments
//Allocate message objects to CAN nodes:
(13) while(CAN_PANCTR.U & (0x00000100 | 0x00000200)); // busy?
(14) CAN_PANCTR = (1 << 24)|(0 << 16)| 2; // MO 0 to list 1
(15) while(CAN_PANCTR.U & (0x00000100 | 0x00000200)); // busy?
(16) CAN_PANCTR = (2 << 24)|(1 << 16)| 2; // MO 1 to list 2
//Init MO_0 (list 1, node 0)
(17) CAN_MOCTR0 = (1<<27)|(1<<26)|(1<<25)|(1<<21); // set to Tx
(18) CAN_MOFCR0 = (8 << 24);              // DLC=8
(19) CAN_MOAR0 =  (1 << 30)|(0xFF << 18); // PRI=01,ID=0xFF
//Init MO_1 (list 2, node 1)
(20) CAN_MOCTR1 = (1 << 23)|(1 << 21);    // set to Rx
(21) CAN_MOFCR1 = (1 << 16);              // RXIE=1
(22) CAN_MOAR1 =  (1 << 30)|(0xFF << 18);  // PRI=01, ID=0xFF
(23) CAN_MOIPR1 = 0x1;          // RXINP=1 -> select INT_O1
(24) SRC_CANINT1 = (1 << 10) | CAN_SRN1INT; // enable INT_01
(25) interruptHandlerInstall (CAN_SRN1INT, & CAN1_Rx_irq);
// Load Data, Start CAN nodes
(26) CAN_MODATAL0 = 0x0D0D0D0D; // load data, lower 4 Bytes
(27) CAN_MODATAH0 = 0x0E0E0E0E; // load data, higher 4 Bytes
(28) CAN_NCR0 &= ~((1<<6)| 1);  // reset CCE and INIT
(29) CAN_NCR1 &= ~((1<<6)| 1);  // reset CCE and INIT
// set transmit request to send message
(30) CAN_MOCTR0 = (1<<24)|(1<<19)|(1<<6);//TXRQ=1,NEWDAT=1
(31) while((CAN_MOSTAT1.B.NEWDAT) != 1);       //check if Rx
```

Note: Before an interrupt is able to occur, the interrupt system has to be globally enabled. The Interrupt Control Register (ICR) holds the global interrupt enable bit (ICR.IE) which enables the CPU service request system. Most compiler support the attribute (or similar):

__enable();
to set this bit. (See also Architecture Manual for more details)

21.6 MultiCAN+ Kernel Registers

This section describes the kernel registers of the MultiCAN+ module. All MultiCAN+ kernel register names described in this section are also referenced in other parts of the TC21x/TC22x/TC23x User's Manual by the module name prefix "CAN_".

MultiCAN+ Kernel Register Overview

The MultiCAN+ Kernel include three blocks of registers:

- Global Module Registers
- Node Registers, for each CAN node x
- Message Object Registers, for each message object n

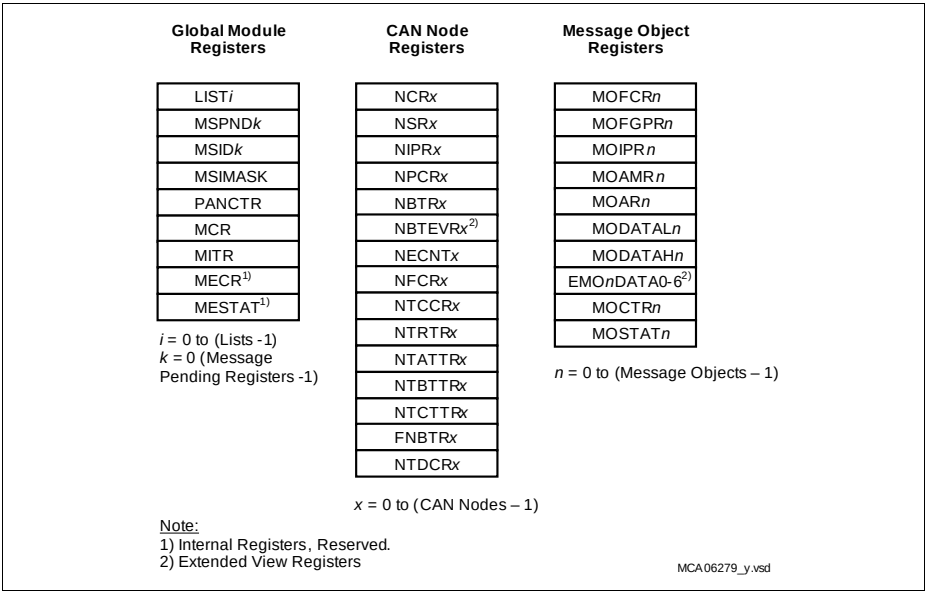


Figure 21-26 MultiCAN+ Kernel Registers

The registers of the MultiCAN+ module kernel are listed below.

Table 21-9 Registers Address Space - MultiCAN+ Kernel Registers

Module	Base Address	End Address	Note
CAN	$F001\ 8000_H$	$F001\ BFFF_H$	-
CAN1	$F002\ 8000_H$	$F002\ BFFF_H$	-

Table 21-10 Registers Overview - MultiCAN+ Kernel Registers

Short Name	Description	Offset Addr[1)]	Access Mode[2)]		Reset	Description see
			Read	Write		
Global Module Registers						
LISTi	List Register i	$0100_H + i \times 4_H$	U, SV	U, SV, P	Application Reset	Page 21-87
MSPNDk	Message Pending Register k	$0140_H + k \times 4_H$	U, SV	U, SV, P	Application Reset	Page 21-89
MSIDk	Message Index Register k	$0180_H + k \times 4_H$	U, SV	U, SV, P	Application Reset	Page 21-90
MSIMASK	Message Index Mask Register	$01C0_H$	U, SV	U, SV, P	Application Reset	Page 21-91
PANCTR	Panel Control Register	$01C4_H$	U, SV	U, SV, P	Application Reset	Page 21-80
MCR	Module Control Register	$01C8_H$	U, SV	U, SV, P	Application Reset	Page 21-84
MITR	Module Interrupt Trigger Reg.	$01CC_H$	U, SV	U, SV, P	Application Reset	Page 21-86
MECR	Measurement Control Register	$01D0_H$	U, SV	U, SV, P	Application Reset	Page 21-92
MESTAT	Measurement Status Register	$01D4_H$	U, SV	U, SV, P	Application Reset	Page 21-94
CAN Node Registers						
NCRx	Node x Control Register	$0200_H + x \times 100_H$	U, SV	U, SV, P	Application Reset	Page 21-95

Table 21-10 Registers Overview - MultiCAN+ Kernel Registers (cont'd)

Short Name	Description	Offset Addr[1)]	Access Mode[2)]		Reset	Description see
			Read	Write		
NSRx	Node x Status Register	$0204_H + x \times 100_H$	U, SV	U, SV, P	Application Reset	Page 21-99
x	Node x Interrupt Pointer Reg.	$0208_H + x \times 100_H$	U, SV	U, SV, P	Application Reset	Page 21-103
NPCRx	Node x Port Control Register	$020C_H + x \times 100_H$	U, SV	U, SV, P	Application Reset	Page 21-105
NBTRx	Node x Bit Timing Register	$0210_H + x \times 100_H$	U, SV	U, SV, P	Application Reset	Page 21-106
NBTEVRx	Node x Bit Timing Extended View Register	$0210_H + x \times 100_H$	U, SV	U, SV, P	Application Reset	Page 21-108
NECNTx	Node x Error Counter Register	$0214_H + x \times 100_H$	U, SV	U, SV, P	Application Reset	Page 21-113
NFCRx	Node x Frame Counter Register	$0218_H + x \times 100_H$	U, SV	U, SV, P	Application Reset	Page 21-114
NTCCRx	Node x Timer Clock Control Register	$021C_H + x \times 100_H$	U, SV	U, SV, P	Application Reset	Page 21-118
NTRTRx	Node x Timer Receive Timeout Register	$0220_H + x \times 100_H$	U, SV	U, SV, P	Application Reset	Page 21-120
NTATTRx	Node x Timer A Transmit Trigger Register	$0224_H + x \times 100_H$	U, SV	U, SV, P	Application Reset	Page 21-122
NTBTTRx	Node x Timer B Transmit Trigger Register	$0228_H + x \times 100_H$	U, SV	U, SV, P	Application Reset	Page 21-123

Table 21-10 Registers Overview - MultiCAN+ Kernel Registers (cont'd)

Short Name	Description	Offset Addr[1)]	Access Mode[2)]		Reset	Description see
			Read	Write		
NTCTTRx	Node x Timer C Transmit Trigger Register	$022C_H + x \times 100_H$	U, SV	U, SV, P	Application Reset	Page 21-124
FNBTRx	Fast Node x Bit Timing Register	$0238_H + x \times 100_H$	U, SV	U, SV, P	Application Reset	Page 21-110
NTDCRx	Node x Transmitter Delay Compensation Register	$023C_H + x \times 100_H$	U, SV	U, SV, P	Application Reset	Page 21-111
Message Object Registers						
MOFCRn	Message Object n Function Control Register	$1000_H + n \times 20_H$	U, SV	U, SV, P	Application Reset	Page 21-135
MOFGPRn	Message Object n FIFO/Gateway Pointer Register	$1004_H + n \times 20_H$	U, SV	U, SV, P	Application Reset	Page 21-141
MOIPRn	Message Object n Interrupt Pointer Register	$1008_H + n \times 20_H$	U, SV	U, SV, P	Application Reset	Page 21-133
MOAMRn	Message Object n Acceptance Mask Register	$100C_H + n \times 20_H$	U, SV	U, SV, P	Application Reset	Page 21-143
MODATALn	Message Object n Data Register Low	$1010_H + n \times 20_H$	U, SV	U, SV, P	Application Reset	Page 21-147
MODATAHn	Message Object n Data Register High	$1014_H + n \times 20_H$	U, SV	U, SV, P	Application Reset	Page 21-148

Table 21-10 Registers Overview - MultiCAN+ Kernel Registers (cont'd)

Short Name	Description	Offset Addr[1]	Access Mode[2]		Reset	Description see
			Read	Write		
EMOnDATA	Extended Message Object n Data Register 0	$1000_H + n \times 20_H$	U, SV	U, SV, P	Application Reset	Page 21-149
	Extended Message Object n Data Register 1	$1004_H + n \times 20_H$	U, SV	U, SV, P	Application Reset	
	Extended Message Object n Data Register 2	$1008_H + n \times 20_H$	U, SV	U, SV, P	Application Reset	
	Extended Message Object n Data Register 3	$100C_H + n \times 20_H$	U, SV	U, SV, P	Application Reset	
	Extended Message Object n Data Register 4	$1010_H + n \times 20_H$	U, SV	U, SV, P	Application Reset	
	Extended Message Object n Data Register 5	$1014_H + n \times 20_H$	U, SV	U, SV, P	Application Reset	
	Extended Message Object n Data Register 6	$1018_H + n \times 20_H$	U, SV	U, SV, P	Application Reset	
MOARn	Message Object n Arbitration Register	$1018_H + n \times 20_H$	U, SV	U, SV, P	Application Reset	Page 21-144
MOCTRn MOSTATn	Message Object n Control Reg. Message Object n Status Reg.	$101C_H + n \times 20_H$	U, SV	U, SV, P	Application Reset	Page 21-125 Page 21-128

1) The absolute register address is calculated as follows:
Module Base Address (**Table 21-9**) + Offset Address (shown in this column)
Further, the following ranges for parameters i, k, x, and n are valid: i = 0-15, k = 0-7, x = 0-2, n = 0-127.

2) Accesses to empty addresses: nBE

List of Access Protection Abbreviations

U - User Mode

SV - Supervisor Mode

BE - Bus Error

nBE - no Bus Error

P - Access Protection, as defined by the ACCEN Register

E - ENDINIT

SE - Safety ENDINIT

Figure 21-27 shows the MultiCAN+ register address map.

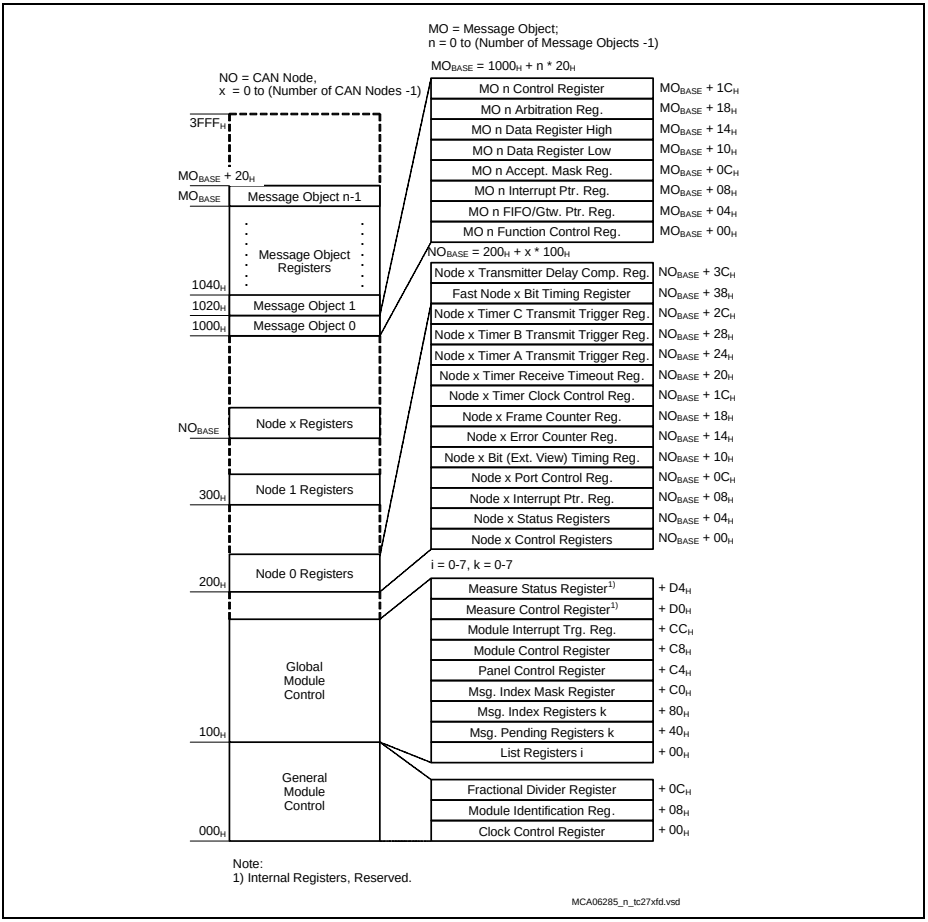


Figure 21-27 MultiCAN+ Register Address Map

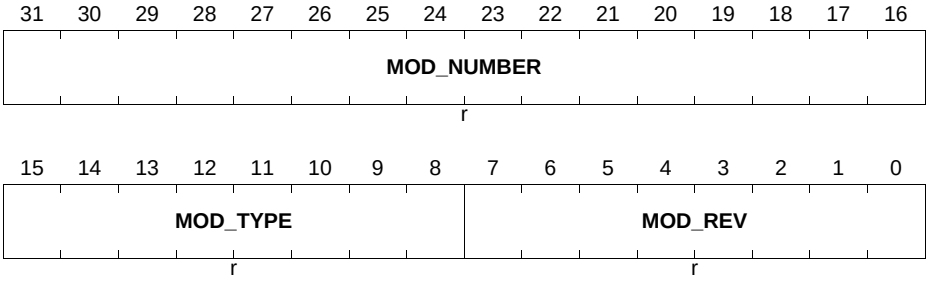
21.6.1 Global Module Registers

All list operations such as allocation, de-allocation and relocation of message objects within the list structure are performed via the Command Panel. It is not possible to modify the list structure directly by software by writing to the message objects and the LIST registers.

Module Identification Register.

ID

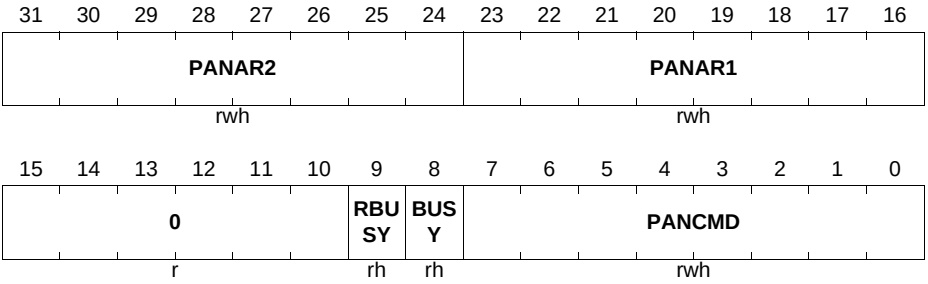
Module Identification Register (008$_H$) Reset Value: 00B5 C0XX$_H$



Field	Bits	Type	Description
MOD_REV	[7:0]	r	**Module Revision Number** MOD_REV defines the revision number. The value of a module revision starts with 01$_H$ (first revision).
MOD_TYPE	[15:8]	r	**Module Type** C0$_H$ Define the module as a 32-bit module.
MOD_NUMBER	[31:16]	r	**Module Number Value** This bit field defines the MultiCAN+ module identification number (=00B5H)

The Panel Control Register PANCTR is used to start a new command by writing the command arguments and the command code into its bit fields.

PANCTR
Panel Control Register **($1C4_H$)** **Reset Value: $0000\ 0301_H$**



Field	Bits	Type	Description
PANCMD	[7:0]	rwh	**Panel Command** This bit field is used to start a new command by writing a panel command code into it. At the end of a panel command, the NOP (no operation) command code is automatically written into PANCMD. The coding of PANCMD is defined in **Table 21-11**.
BUSY	8	rh	**Panel Busy Flag** 0_B Panel has finished command and is ready to accept a new command. 1_B Panel operation is in progress.
RBUSY	9	rh	**Result Busy Flag** 0_B No update of PANAR1 and PANAR2 is scheduled by the list controller. 1_B A list command is running (BUSY = 1) that will write results to PANAR1 and PANAR2, but the results are not yet available.
PANAR1	[23:16]	rwh	**Panel Argument 1** See **Table 21-11**.
PANAR2	[31:24]	rwh	**Panel Argument 2** See **Table 21-11**.
0	[15:10]	r	**Reserved** Read as 0; should be written with 0.

Panel Commands

A panel operation consists of a command code (PANCMD) and up to two panel arguments (PANAR1, PANAR2). Commands that have a return value deliver it to the PANAR1 bit field. Commands that return an error flag deliver it to bit 31 of the Panel Control Register, this means bit 7 of PANAR2.

Table 21-11 Panel Commands

PANCMD	PANAR2	PANAR1	Command Description
00_H	–	–	**No Operation** Writing 00_H to PANCMD has no effect. No new command is started.
01_H	**Result:** Bit 7: ERR Bit 6-0: undefined	–	**Initialize Lists** Run the initialization sequence to reset the CTRL and LIST fields of all message objects. List registers LIST[7:0] are set to their reset values. This results in the de-allocation of all message objects. The initialization command requires that bits NCRx.INIT and NCRx.CCE are set for all CAN nodes. Bit 7 of PANAR2 (ERR) reports the success of the operation: 0_B Initialization was successful 1_B Not all NCRx.INIT and NCRx.CCE bits are set. Therefore, no initialization is performed. The initialize lists command is automatically performed with each reset of the MultiCAN+ module, but with the exception that all message object registers are reset, too.
02_H	**Argument:** List Index	**Argument:** Message Object Number	**Static Allocate** Allocate message object to a list. The message object is removed from the list that it currently belongs to, and appended to the end of the list, given by PANAR2. This command is also used to deallocate a message object. In this case, the target list is the list of unallocated elements (PANAR2 = 0).

Table 21-11 Panel Commands (cont'd)

PANCMD	PANAR2	PANAR1	Command Description
03_H	**Argument:** List Index **Result:** Bit 7: ERR Bit 6-0: undefined	**Result:** Message Object Number	**Dynamic Allocate** Allocate the first message object of the list of unallocated objects to the selected list. The message object is appended to the end of the list. The message number of the message object is returned in PANAR1. An ERR bit (bit 7 of PANAR2) reports the success of the operation: 0_B Success. 1_B The operation has not been performed because the list of unallocated elements was empty.
04_H	**Argument:** Destination Object Number	**Argument:** Source Object Number	**Static Insert Before** Remove a message object (source object) from the list that it currently belongs to, and insert it before a given destination object into the list structure of the destination object. The source object thus becomes the predecessor of the destination object.
05_H	**Argument:** Destination Object Number **Result:** Bit 7: ERR Bit 6-0: undefined	**Result:** Object Number of inserted object	**Dynamic Insert Before** Insert a new message object before a given destination object. The new object is taken from the list of unallocated elements (the first element is chosen). The number of the new object is delivered as a result to PANAR1. An ERR bit (bit 7 of PANAR2) reports the success of the operation: 0_B Success. 1_B The operation has not been performed because the list of unallocated elements was empty.

Table 21-11 Panel Commands (cont'd)

PANCMD	PANAR2	PANAR1	Command Description
06_H	**Argument:** Destination Object Number	**Argument:** Source Object Number	**Static Insert Behind** Remove a message object (source object) from the list that it currently belongs to, and insert it behind a given destination object into the list structure of the destination object. The source object thus becomes the successor of the destination object.
07_H	**Argument:** Destination Object Number **Result:** Bit 7: ERR Bit 6-0: undefined	**Result:** Object Number of inserted object	**Dynamic Insert Behind** Insert a new message object behind a given destination object. The new object is taken from the list of unallocated elements (the first element is chosen). The number of the new object is delivered as result to PANAR1. An ERR bit (bit 7 of PANAR2) reports the success of the operation: 0_B Success. 1_B The operation has not been performed because the list of unallocated elements was empty.
08_H - FF_H	–	–	**Reserved**

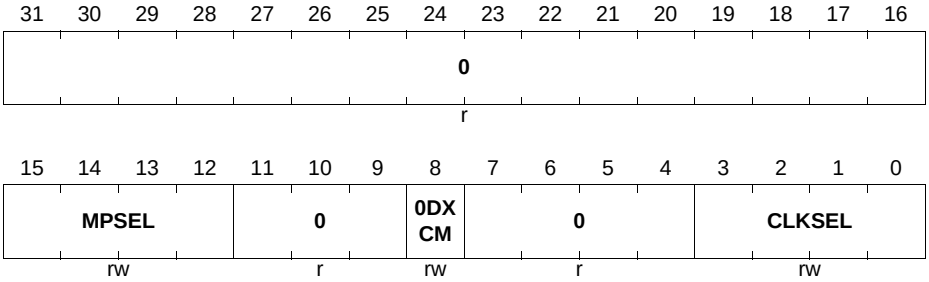
The Module Control Register MCR contains basic settings that determine the operation of the MultiCAN+ module.

The write access to the lowest byte of the MCR register is possible only if the CCE bits of all CAN nodes are set (NCRx.CCE bits). The NCRx.INIT bits will be automatically set when the lowest byte of the MCR register is written, independent of the setting of the CCE bits. The INIT bits have to be reset by software in order to activate the CAN nodes.

The reconfiguration of the clock source has to be done by using two writes: first a write of zero to the CLKSEL bit field, and then a second write defining the new clock source. Between the first and the second write a delay of 4 / f_A + 2 / f_{CAN} number of cycles must be inserted by software, where f_A is the frequency being switched off with the first write. Exception: in case that f_{ASYN_CAN} is selected as the baud rate logic clock (MCR.CLKSEL = 1), no delay cycles between the writes are necessary. In both cases, simply using one write defining the new clock source is not allowed.

Note: If the baud rate logic is supplied from an unstable clock source, or no clock at all, the CAN functionality is not guaranteed.

MCR
Module Control Register (1C8$_H$) Reset Value: 0000 0000$_H$

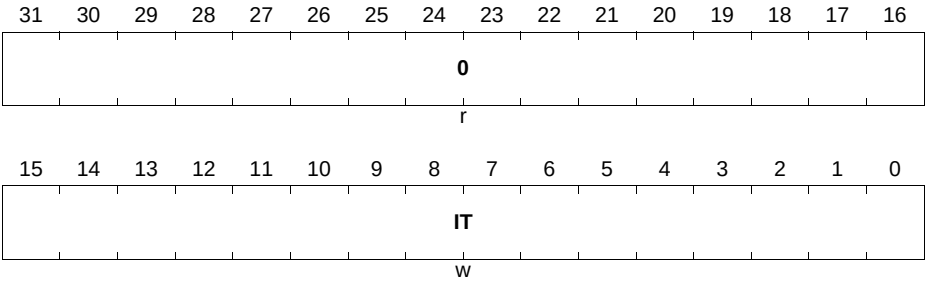


Field	Bits	Type	Description
CLKSEL	[3:0]	rw	**Baud Rate Logic Clock Select** 0000_B No clock supplied 0001_B f_{ASYN_CAN} 0010_B Oscillator Clock 0100_B E-Ray clock (f_{ERAY}) 1000_B hard wired to 0 $...{}_B$ not allowed

Field	Bits	Type	Description
DXCM	8	rw	**Debug Over CAN Messages Enable** This bit enables the debug over serial connections between DAP and CAN module. 0_B DXCM disabled 1_B DXCM enabled If enabled the highest two message objects are reserved for debugger communication. DXCM is described in detail in the OCDS chapter.
0	8	rw	**Reserved** Read as 0; should be written with 0.
MPSEL	[15:12]	rw	**Message Pending Selector** Bit field MPSEL makes it possible to select the bit position of the message pending bit after a message reception/transmission by a mixture of the MOIPRn register bit fields RXINP, TXINP, and MPN. Selection details are given in **Figure 21-21** on **Page 21-48**.
0	[31:16], [11:9], [7:4]	r	**Reserved** Read as 0; should be written with 0.

The Interrupt Trigger Register ITR is used to trigger interrupt requests on each interrupt output line by software.

MITR
Module Interrupt Trigger Register (1CC$_H$) **Reset Value: 0000 0000$_H$**



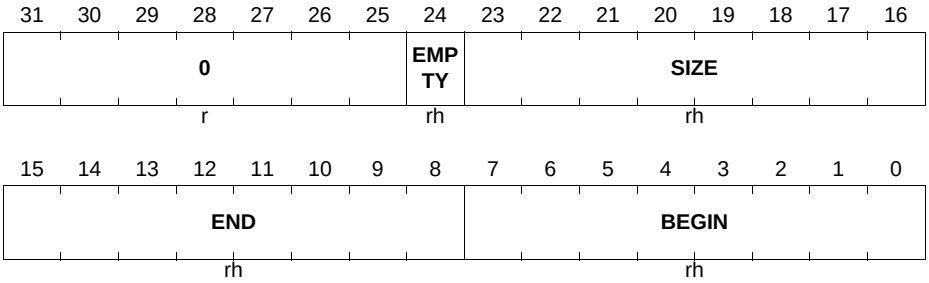
Field	Bits	Type	Description
IT	[15:0]	w	**Interrupt Trigger** Writing a 1 to IT[m] (m = 0-15) generates an interrupt request on interrupt output line INT_O[m]. Writing a 0 to IT[m] has no effect. Bit field IT is always read as 0. Multiple interrupt requests can be generated with a single write operation to MITR by writing a 1 to several bit positions of IT.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

List Pointer and List Register

Each CAN node has a list that determines the allocated message objects. Additionally, a list of all unallocated objects is available. Furthermore, general purpose lists are available which are not associated to a CAN node. The List Registers are assigned in the following way:

- LIST0 provides the list of all unallocated objects
- LIST1 provides the list for CAN node 0
- LIST2 provides the list for CAN node 1
- ...
- LIST3 provides the list for CAN node 2

LIST0
List Register 0 **(100_H)** **Reset Value: 007F 7F00$_H$**
LISTn (n = 1-15)
List Register n **(100_H+n*4$_H$)** **Reset Value: 0100 0000$_H$**



Field	Bits	Type	Description
BEGIN	[7:0]	rh	**List Begin** BEGIN indicates the number of the first message object in list i.
END	[15:8]	rh	**List End** END indicates the number of the last message object in list i.
SIZE	[23:16]	rh	**List Size** SIZE indicates the number of elements in the list i. SIZE = number of list elements - 1 SIZE = 0 indicates that list i is empty.

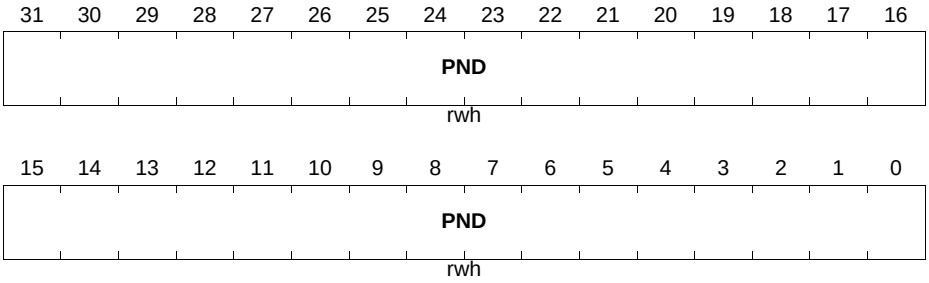
Field	Bits	Type	Description
EMPTY	24	rh	**List Empty Indication** 0_B At least one message object is allocated to list i. 1_B No message object is allocated to the list i. List i is empty.
0	[31:25]	r	**Reserved** Read as 0.

Message Notifications

When a message object n generates an interrupt request upon the transmission or reception of a message, then the request is routed to the interrupt output line selected by the bit field MOIPRn.TXINP or MOIPRn.RXINP of the message object n. As there are more message objects than interrupt output lines, an interrupt routine typically processes requests from more than one message object. Therefore, a priority selection mechanism is implemented in the MultiCAN+ module to select the highest priority object within a collection of message objects.

The Message Pending Register MSPNDk contains the pending interrupt notification of list i.

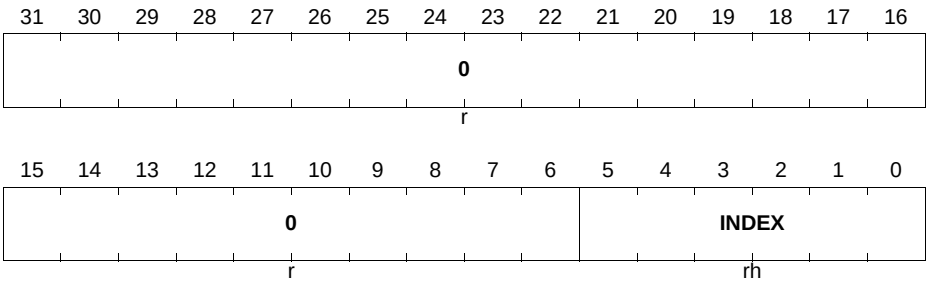
MSPNDk (k = 0-7)
Message Pending Register k **(140_H+k*4_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
PND	[31:0]	rwh	**Message Pending** When a message interrupt occurs, the message object sets a bit in one of the MSPND register, where the bit position is given by the MPN[4:0] field of the IPR register of the message object. The register selection n is given by the higher bits of MPN. The register bits can be cleared by software (write 0). Writing a 1 has no effect.

Each Message Pending Register has a Message Index Register MSIDk associated with it. The Message Index Register shows the active (set) pending bit with lowest bit position within groups of pending bits.

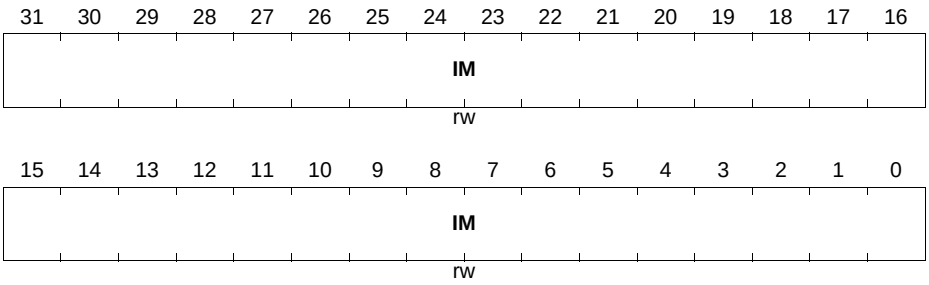
MSIDk (k = 0-7)
Message Index Register k **(180_H+k*4_H)** **Reset Value: 0000 0020_H**



Field	Bits	Type	Description
INDEX	[5:0]	rh	**Message Pending Index** The value of INDEX is given by the bit position i of the pending bit of MSPNDk with the following properties: 1. MSPNDk[i] & IM[i] = 1 2. i = 0 or MSPNDk[i-1:0] & IM[i-1:0] = 0 If no bit of MSPNDk satisfies these conditions then INDEX reads 100000_B. Thus INDEX shows the position of the first pending bit of MSPNDk, in which only those bits of MSPNDk that are selected in the Message Index Mask Register are taken into account.
0	[31:6]	r	**Reserved** Read as 0; should be written with 0.

The Message Index Mask Register MSIMASK selects individual bits for the calculation of the Message Pending Index. The Message Index Mask Register is used commonly for all Message Pending registers and their associated Message Index registers.

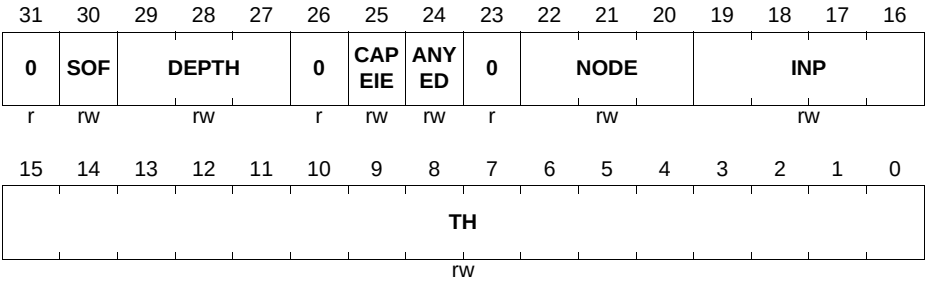
MSIMASK
Message Index Mask Register **($1C0_H$)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
IM	[31:0]	rw	**Message Index Mask** Only those bits in MSPNDk for which the corresponding Index Mask bits are set contribute to the calculation of the Message Index.

The Measurement Control Register MECR controls the CAN edge timing measurement function for calibration purposes.

MECR
Measurement Control Register (1D0$_H$) Reset Value: 0000 0000$_H$

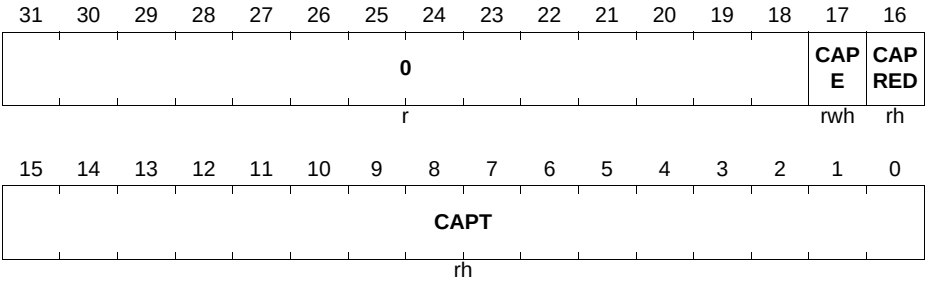


Field	Bits	Type	Description
TH	[15:0]	rw	**Threshold** This bit field contains the threshold value for the measurement timer. If TH = 0000$_H$, the timer is stopped and the capture function is disabled.
INP	[19:16]	rw	**Interrupt Node Pointer** INP selects the interrupt output line INT_Om (m = 0-15) for a capture event interrupt. 0000$_B$ Interrupt output line INT_O0 is selected. 0001$_B$ Interrupt output line INT_O1 is selected. …$_B$ … 1110$_B$ Interrupt output line INT_O14 is selected. 1111$_B$ Interrupt output line INT_O15 is selected.
NODE	[22:20]	rw	**Node** This bit field determines the CAN node x whose input line RXDCANx is used for start and capture of the measurement timer. 000$_B$ Node 0 001$_B$ Node 1 …$_B$ … 010$_B$ Node 2

Field	Bits	Type	Description
ANYED	24	rw	**Any Edge** This bit enables capture on any edge of CAN input line specified by NODE. 0_B Capture on falling (dominant) edge only 1_B Capture on rising (recessive) or falling (dominant) edge
CAPEIE	25	rw	**Capture Event Interrupt Enable** This bit enables the capture event interrupt. 0_B Capture event interrupt is disabled 1_B Capture event interrupt is enabled Bit field INP selects the interrupt output line which becomes activated at this type of interrupt.
DEPTH	[29:27]	rw	**Digital Glitch Filter Depth** DEPTH determines the number of input samples clocked with f_{CLC} that are taken into account for the calculation of the floating average. The higher DEPTH is chosen to be, the longer the glitches that are suppressed and the longer the delay of the input signal introduced by this filter. 000_B off, default 001_B Filter depth of 8 cycles 010_B Filter depth of 16 cycles 011_B Filter depth of 32 cycles 100_B Filter depth of 64 cycles 101_B Filter depth of 128 cycles 110_B Filter depth of 255 cycles 111_B not allowed, reserved
SOF	30	rw	**Start Of Frame** This bit selects falling edge or any edge as measurement for start of frame detection. 0_B Measurement starts with any falling edge 1_B Measurement starts with falling Start of Frame edge. i.e any falling edge that occurs while the CAN node is in idle state
0	23, 26, 31	r	**Reserved** Read as 0; should be written with 0.

The Measurement Status Register MESTAT contains the status information of the CAN edge timing measurement.

MESTAT
Measurement Status Register (1D4$_H$) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
CAPT	[15:0]	rh	**Captured Timer** This bit field contains the captured measurement timer content. The timer itself is cleared and started by the first falling (dominant) edge of a CAN frame on the input line of the CAN node specified by MECR.NODE. The timer is incremented by the module control clock f_{CLC} and will be stopped when FFFF$_H$ is reached. If MECR.TH = 0000$_H$, the timer is always stopped. A capture will take place if all the following conditions are met: 1. MECR.TH > 0000$_H$ 2. Timer is cleared and started by new frame 3. Timer reaches MECR.TH 4. This node is not sending and first edge (as specified by MECR.ANYED) after 3. occurs on input line Capture will be repeated for the following CAN frames until MECR.TH is cleared.
CAPRED	16	rh	**Captured Rising Edge** This bit indicates the type of edge that caused the last capture event. 0_B Capture occurred on falling (dominant) edge 1_B Capture occurred on rising (recessive) edge

Field	Bits	Type	Description
CAPE	17	rwh	**Capture Event** This flag is set on a capture event. It must be reset by software. 0_B No capture event has occurred since last flag reset 1_B Capture event has occurred since last flag reset An interrupt request is generated if MECR.CAPEIE = 1. If CAPE=1 then no further measurement results are posted to MESTAT.CAPT and MESTAT.CAPRED. CAPE bit has to be cleared to re-enable update of MESTAT.CAPT and MESTAT.CAPRED.
0	[31:18]	r	**Reserved** Read as 0; should be written with 0.

21.6.2 CAN Node Registers

The CAN node registers are built in for each CAN node of the MultiCAN+ module. They contain information that is directly related to the operation of the CAN nodes and are shared among the nodes.

The Node Control Register contains basic settings that determine the operation of the CAN node.

CAN_NCRx (x = 0-2)
Node x Control Register **(200_H+x*100_H)** **Reset Value: 0000 0041_H**
CAN1_NCRy (y=0-2)
Node y Control Register **(200_H+y*100_H)** **Reset Value: 0000 0041_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						FDEN	SUSEN	CALM	CCE	TXDIS	CANDIS	ALIE	LECIE	TRIE	INIT
r						rw	rw	rw	rw	rw	rw	rw	rw	rw	rwh

Field	Bits	Type	Description
INIT	0	rwh	**Node Initialization** 0_B Resetting bit INIT enables the participation of the node in the CAN traffic. If the CAN node is in the bus-off state, the ongoing bus-off recovery (which does not depend on the INIT bit) is continued. With the end of the bus-off recovery sequence the CAN node is allowed to take part in the CAN traffic. If the CAN node is not in the bus-off state, a sequence of 11 consecutive recessive bits must be detected before the node is allowed to take part in the CAN traffic. 1_B Setting this bit terminates the participation of this node in the CAN traffic. Any ongoing frame transfer is cancelled and the transmit line goes recessive. If the CAN node is in the bus-off state, then the running bus-off recovery sequence is continued. If the INIT bit is still set after the successful completion of the bus-off recovery sequence, i.e. after detecting 128 sequences of 11 consecutive recessive bits (11×1), then the CAN node leaves the bus-off state but remains inactive as long as INIT remains set. Bit INIT is automatically set when the CAN node enters the bus-off state (see **Page 21-30**).
TRIE	1	rw	**Transfer Interrupt Enable** TRIE enables the transfer interrupt of CAN node x. This interrupt is generated after the successful reception or transmission of a CAN frame in node x. 0_B Transfer interrupt is disabled. 1_B Transfer interrupt is enabled. Bit field NIPRx.TRINP selects the interrupt output line which becomes activated at this type of interrupt.

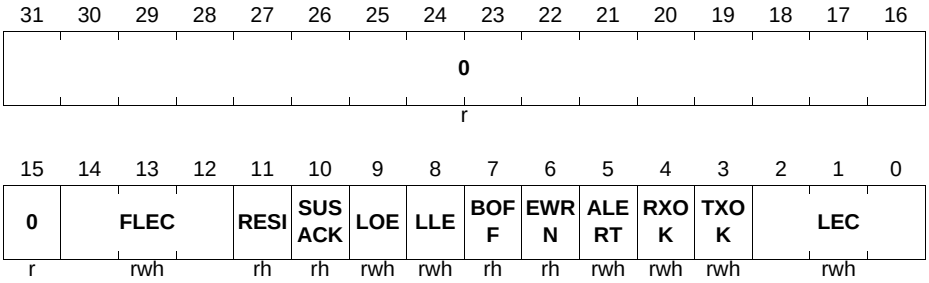
Field	Bits	Type	Description
LECIE	2	rw	**LEC Indicated Error Interrupt Enable** LECIE enables the last error code interrupt of CAN node x. This interrupt is generated with each hardware update of bit field NSRx.LEC with LEC > 0 (CAN protocol error). 0_B Last error code interrupt is disabled. 1_B Last error code interrupt is enabled. Bit field NIPRx.LECINP selects the interrupt output line which becomes activated at this type of interrupt.
ALIE	3	rw	**Alert Interrupt Enable** ALIE enables the alert interrupt of CAN node x. This interrupt is generated by any one of the following events: • A change of bit NSRx.BOFF • A change of bit NSRx.EWRN • A List Length Error, which also sets bit NSRx.LLE • A List Object Error, which also sets bit NSRx.LOE 0_B Alert interrupt is disabled. 1_B Alert interrupt is enabled. Bit field NIPRx.ALINP selects the interrupt output line which becomes activated at this type of interrupt.
CANDIS	4	rw	**CAN Disable** Setting this bit disables the CAN node. The CAN node first waits until it is bus-idle or bus-off. Then bit NSRx.SUSACK and NCRx.INIT is automatically set, and an alert interrupt is generated if bit ALIE is set.
TXDIS	5	rw	**Transmit Disable** Setting this bit disables the transmission on CAN node x as soon as bus-idle is reached. Reception and bits in MOSTATn, e.g. TXRQ, will not be influenced.
CCE	6	rw	**Configuration Change Enable** 0_B The Bit Timing Register, the Port Control Register, Error Counter Register and NCRx.FDEN bit may only be read. All attempts to modify them are ignored. 1_B The Bit Timing Register, the Port Control Register, Error Counter Register and NCRx.FDEN bit may be read and written.

Field	Bits	Type	Description
CALM	7	rw	**CAN Analyzer Mode** If this bit is set, then the CAN node operates in Analyzer Mode. This means that messages may be received, but not transmitted. No acknowledge is sent on the CAN bus upon frame reception. Active-error flags are sent recessive instead of dominant. The transmit line is continuously held at recessive (1) level. Bit CALM can be written only while bit INIT is set.
SUSEN	8	rw	**Suspend Enable** This bit makes it possible to set the CAN node into Suspend Mode via OCDS (on chip debug support): 0_B An OCDS suspend trigger is ignored by the CAN node. 1_B An OCDS suspend trigger disables the CAN node: As soon as the CAN node becomes bus-idle or bus-off, bit INIT is internally forced to 1 to disable the CAN node. The actual value of bit INIT remains unchanged. Bit SUSEN is reset via OCDS Reset.
FDEN	9	rw	**CAN Flexible Data-Rate Enable** This bit enables the CAN FD feature: 0_B CAN FD disabled. Message transfer for CAN frames using classical CAN Format. 1_B CAN FD enabled. Message transfer for CAN frames using CAN FD Format[1)]. Bit FDEN is protected by CCE and INIT bit. Please see **Section 21.4.12.4** and **Table 21-18**
0	[31:10]	r	**Reserved** Read as 0; should be written with 0.

1) Message transmission in Classical CAN, Long Frame or Long + Fast CAN FD Frames. Message Reception according to CAN FD Protocol Specification, (i.e Able to Receive Classical CAN Format Frames ISO11898-1 Long Frame and Long + Fast CAN FD Frames)

The Node Status Register NSRx reports errors as well as successfully transferred CAN frames.

CAN_NSRx (x = 0-2)
Node x Status Register **(204_H+x*100_H)** **Reset Value: 0000 0000_H**
CAN1_NSRy (y=0-2)
Node y Status Register **(204_H+y*100_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
LEC	[2:0]	rwh	**Last Error Code** This bit field indicates the type of the last (most recent) CAN error. The encoding of this bit field is described in **Table 21-12**.
TXOK	3	rwh	**Message Transmitted Successfully** 0_B No successful transmission since last (most recent) flag reset. 1_B A message has been transmitted successfully (error-free and acknowledged by at least another node). TXOK must be reset by software (write 0). Writing 1 has no effect.
RXOK	4	rwh	**Message Received Successfully** 0_B No successful reception since last (most recent) flag reset. 1_B A message has been received successfully. RXOK must be reset by software (write 0). Writing 1 has no effect.

Field	Bits	Type	Description
ALERT	5	rwh	**Alert Warning** The ALERT bit is set upon the occurrence of one of the following events (the same events which also trigger an alert interrupt if ALIE is set): • A change of bit NSRx.BOFF • A change of bit NSRx.EWRN • A List Length Error, which also sets bit NSRx.LLE • A List Object Error, which also sets bit NSRx.LOE ALERT must be reset by software (write 0). Writing 1 has no effect.
EWRN	6	rh	**Error Warning Status** 0_B No warning limit exceeded. 1_B One of the error counters REC or TEC reached the warning limit EWRNLVL.
BOFF	7	rh	**Bus-off Status** 0_B CAN controller is not in the bus-off state. 1_B CAN controller is in the bus-off state.
LLE	8	rwh	**List Length Error** 0_B No List Length Error since last (most recent) flag reset. 1_B A List Length Error has been detected during message acceptance filtering. The number of elements in the list that belongs to this CAN node differs from the list SIZE given in the list termination pointer. LLE must be reset by software (write 0). Writing 1 has no effect.
LOE	9	rwh	**List Object Error** 0_B No List Object Error since last (most recent) flag reset. 1_B A List Object Error has been detected during message acceptance filtering. A message object with wrong LIST index entry in the Message Object Status Register has been detected. LOE must be reset by software (write 0). Writing 1 has no effect.

Field	Bits	Type	Description
SUSACK	10	rh	**Suspend Acknowledge** 0_B The CAN node is not in Suspend Mode or a suspend request is pending, but the CAN node has not yet reached bus-idle or bus-off. 1_B The CAN node is in Suspend Mode: The CAN node is inactive (bit NCR.INIT internally forced to 1) due to an OCDS suspend request.
RESI	11	rh	**Received Error State Indicator Flag** This bit is an error flag that is set when the ESI flag in a received CAN FD frame is set. 0_B Last received CAN FD message did not have its ESI flag set. 1_B Last received CAN FD message had its ESI flag set.
FLEC	[14:12]	rwh	**Fast Last Error Code** This bit field indicates the type of the last (most recent) CAN error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC. The encoding of this bit field is described in **Table 21-12**. This field will be cleared to zero when a CAN FD frame with its BRS flag set has been transferred (reception or transmission) without error.
0	[31:15]	r	**Reserved** Read as 0; should be written with 0.

Encoding of the LEC Bit field

Table 21-12 Encoding of the LEC Bit field

LEC Value	Signification
000_B	**No Error:** No error was detected for the last (most recent) message on the CAN bus.
001_B	**Stuff Error:** More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
010_B	**Form Error:** A fixed format part of a received frame has the wrong format.

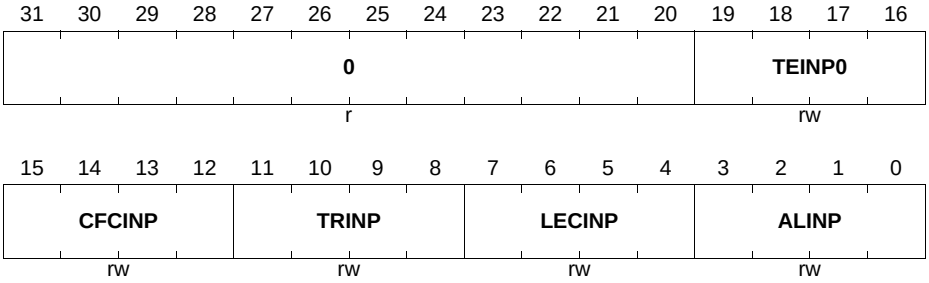
Table 21-12 Encoding of the LEC Bit field (cont'd)

LEC Value	Signification
011_B	**Ack Error:** The transmitted message was not acknowledged by another node.
100_B	**Bit1 Error:** During a message transmission, the CAN node tried to send a recessive level (1) outside the arbitration field and the acknowledge slot, but the monitored bus value was dominant.
101_B	**Bit0 Error:** Two different conditions are signaled by this code: 1. During transmission of a message (or acknowledge bit, active-error flag, overload flag), the CAN node tried to send a dominant level (0), but the monitored bus value was recessive. 2. During bus-off recovery, this code is set each time a sequence of 11 recessive bits has been monitored. The CPU may use this code as indication that the bus is not continuously disturbed.
110_B	**CRC Error:** The CRC checksum of the received message was incorrect.
111_B	**CPU write to LEC:** Whenever the CPU writes the value 111 to LEC, it takes the value 111. Whenever the CPU writes another value to LEC, the written LEC value is ignored.

Note: When a frame in CAN FD format reached data phase with BRS flag set, the next CAN event (error/valid frame) will be shown in FLEC instead of LEC. An error in a fixed stuff bit of CAN FD CRC will be shown as Form and not Stuff Error. Correspondingly CAN event (error/valid frame) will be shown back at LEC after the first bit of the CRC delimiter when CAN FD switches from Data bit rate to Nominal bit rate.

The four five interrupt pointers in the Node Interrupt Pointer Register NIPRx select one out of the sixteen interrupt outputs individually for each type of CAN node interrupt. See also **Page 21-34** for more CAN node interrupt details.

CAN_NIPRx (x = 0-2)
Node x Interrupt Pointer Register (208_H+x*100_H) **Reset Value: 0000 0000_H**
CAN1_NIPRy (y=0-2)
Node y Interrupt Pointer Register (208_H+y*100_H) **Reset Value: 0000 0000_H**

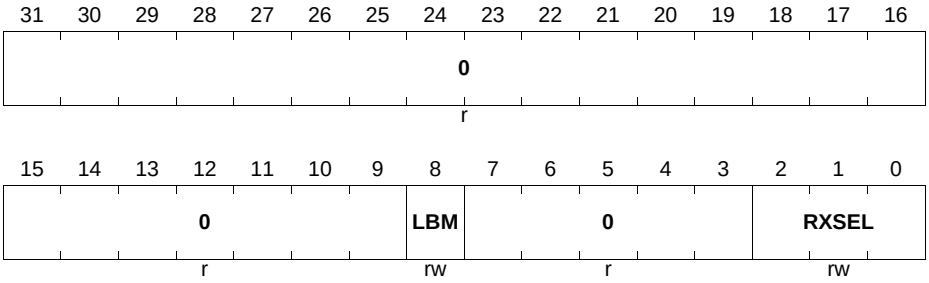


Field	Bits	Type	Description
ALINP	[3:0]	rw	**Alert Interrupt Node Pointer** ALINP selects the interrupt output line INT_Om (m = 0-15), CAN and (m = 0-7), CAN1 for an alert interrupt of CAN Node x. Option 1000_B to 1111_B is Reserved and not applicable for CAN1. 0000_B Interrupt output line INT_O0 is selected. 0001_B Interrupt output line INT_O1 is selected. $\ldots_B$ … 1110_B Interrupt output line INT_O14 is selected. 1111_B Interrupt output line INT_O15 is selected.
LECINP	[7:4]	rw	**Last Error Code Interrupt Node Pointer** LECINP selects the interrupt output line INT_Om (m = 0-15), CAN and (m = 0-7), CAN1 for an LEC interrupt of CAN Node x. Option 1000_B to 1111_B is Reserved and not applicable for CAN1. 0000_B Interrupt output line INT_O0 is selected. 0001_B Interrupt output line INT_O1 is selected. $\ldots_B$ … 1110_B Interrupt output line INT_O14 is selected. 1111_B Interrupt output line INT_O15 is selected.

Field	Bits	Type	Description
TRINP	[11:8]	rw	**Transfer OK Interrupt Node Pointer** TRINP selects the interrupt output line INT_Om (m = 0-15), CAN and (m = 0-7), CAN1 for a transfer OK interrupt of CAN Node x. Option 1000_B to 1111_B is Reserved and not applicable for CAN1. 0000_B Interrupt output line INT_O0 is selected. 0001_B Interrupt output line INT_O1 is selected. $\ldots_B$ … 1110_B Interrupt output line INT_O14 is selected. 1111_B Interrupt output line INT_O15 is selected.
CFCINP	[15:12]	rw	**Frame Counter Interrupt Node Pointer** CFCINP selects the interrupt output line INT_Om (m = 0-15), CAN and (m = 0-7), CAN1 for a frame counter overflow interrupt of CAN Node x. Option 1000_B to 1111_B is Reserved and not applicable for CAN1. 0000_B Interrupt output line INT_O0 is selected. 0001_B Interrupt output line INT_O1 is selected. $\ldots_B$ … 1110_B Interrupt output line INT_O14 is selected. 1111_B Interrupt output line INT_O15 is selected.
TEINP	[19:16]	rw	**Timer Event Interrupt Node Pointer** TEINP selects the interrupt output line INT_Om (m = 0-15), CAN and (m = 0-7), CAN1 for a timer event interrupt of CAN Node x. Option 1000_B to 1111_B is Reserved and not applicable for CAN1. 0000_B Interrupt output line INT_O0 is selected. 0001_B Interrupt output line INT_O1 is selected. $\ldots_B$ … 1110_B Interrupt output line INT_O14 is selected. 1111_B Interrupt output line INT_O15 is selected.
0	[31:20 16]	r	**Reserved** Read as 0; should be written with 0.

The Node Port Control Register NPCRx configures the CAN bus transmit/receive ports. NPCRx can be written only if bit NCRx.CCE is set.

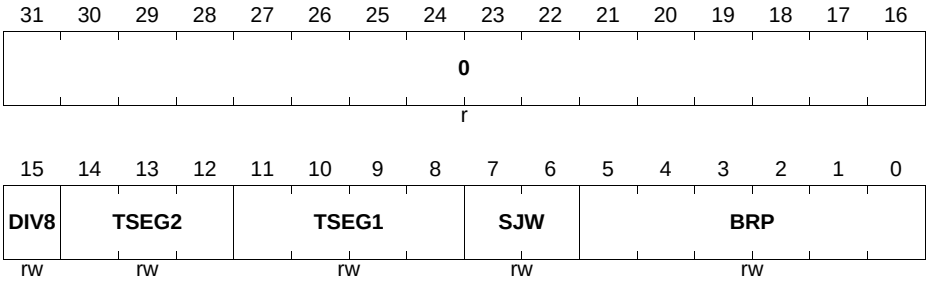
CAN_NPCRx (x = 0-2)
Node x Port Control Register **($20C_H$+x*100_H)** **Reset Value: 0000 0000_H**
CAN1_**NPCRy (y=0-2)**
Node y Port Control Register **($20C_H$+y*100_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
RXSEL	[2:0]	rw	**Receive Select** RXSEL selects one out of 8 possible receive inputs. The CAN receive signal is performed only through the selected input. *Note: In TC21x/TC22x/TC23x, only specific combinations of RXSEL are available (see also **"Node Receive Input Selection" on Page 21-170** for description and the page before for RXSEL selections).*
LBM	8	rw	**Loop-Back Mode** 0_B Loop-Back Mode is disabled. 1_B Loop-Back Mode is enabled. This node is connected to an internal (virtual) loop-back CAN bus. All CAN nodes which are in Loop-Back Mode are connected to this virtual CAN bus so that they can communicate with each other internally. The external transmit line is forced recessive in Loop-Back Mode.
0	[7:3], [31:9]	r	**Reserved** Read as 0; should be written with 0.

The Node Bit Timing Register NBTRx contains all parameters to set up the bit timing for the CAN transfer. NBTRx can be written only if bit NCRx.CCE is set. Please note that NBTRx is a register with two views, depending on NCRx.FDEN settings different view applies. (i.e When NCRx.FDEN=0, CAN_NBTRx view applies and when NCRx.FDEN=1, CAN_NBTEVRx applies)

CAN_NBTRx (x = 0-2)
Node x Bit Timing Register **(210_H+x*100_H)** **Reset Value: 0000 0000_H**
CAN1_**NBTRy (y=0-2)**
Node y Bit Timing Register **(210_H+y*100_H)** **Reset Value: 0000 0000_H**

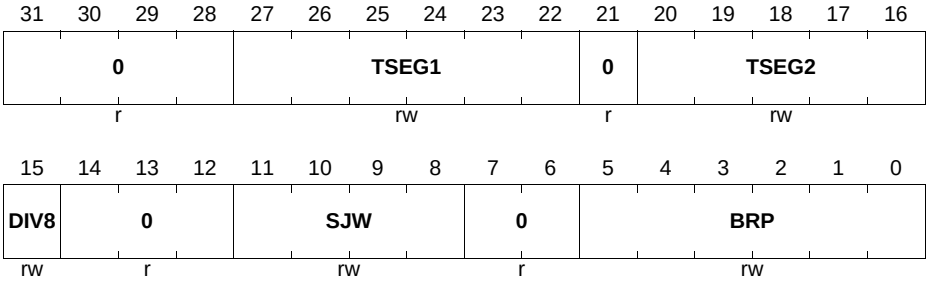


Field	Bits	Type	Description
BRP	[5:0]	rw	**Baud Rate Prescaler** The duration of one time quantum is given by (BRP + 1) clock cycles if DIV8 = 0. The duration of one time quantum is given by 8 × (BRP + 1) clock cycles if DIV8 = 1.
SJW	[7:6]	rw	**(Re) Synchronization Jump Width** (SJW + 1) time quanta are allowed for re-synchronization.
TSEG1	[11:8]	rw	**Time Segment Before Sample Point** (TSEG1 + 1) time quanta is the user-defined nominal time between the end of the synchronization segment and the sample point. It includes the propagation segment, which takes into account signal propagation delays. The time segment may be lengthened due to re-synchronization. Valid values for TSEG1 are 2 to 15.

Field	Bits	Type	Description
TSEG2	[14:12]	rw	**Time Segment After Sample Point** (TSEG2 + 1) time quanta is the user-defined nominal time between the sample point and the start of the next synchronization segment. It may be shortened due to re-synchronization. Valid values for TSEG2 are 1 to 7.
DIV8	15	rw	**Divide Prescaler Clock by 8** 0_B A time quantum lasts (BRP+1) clock cycles. 1_B A time quantum lasts 8 × (BRP+1) clock cycles.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

The Node Bit Timing Extended View Register NBTEVRx is applicable only when NCRx.FDEN=1 (CAN FD enabled). NBTEVRx contains all parameters to set up CAN bit timing for Nominal Bit Rate. NBTRx register can be written only if bit NCRx.CCE is set.

CAN_NBTEVRx (x = 0-2)
Node x Bit Timing Extended View Register(210_H+x*100_H) Reset Value: 0000 0000_H
CAN1_NBTEVRy (y = 0-2)
Node y Bit Timing Extended View Register(210_H+y*100_H) Reset Value: 0000 0000_H

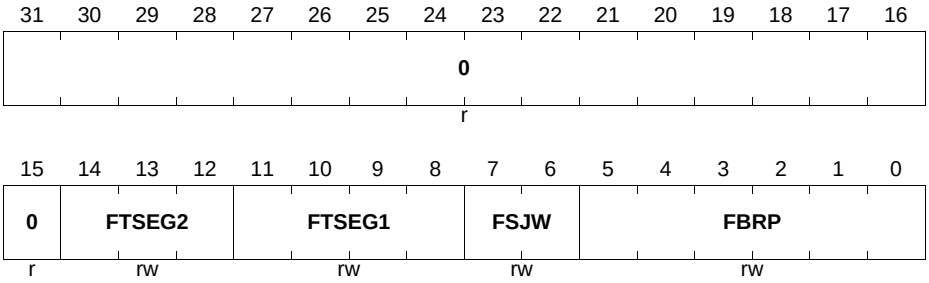


Field	Bits	Type	Description
BRP	[5:0]	rw	**Baud Rate Prescaler** The duration of one time quantum is given by (BRP + 1) clock cycles if DIV8 = 0. The duration of one time quantum is given by 8 × (BRP + 1) clock cycles if DIV8 = 1.
SJW	[11:8]	rw	**(Re) Synchronization Jump Width** (SJW + 1) time quanta are allowed for re-synchronization.
DIV8	15	rw	**Divide Prescaler Clock by 8** 0_B A time quantum lasts (BRP+1) clock cycles. 1_B A time quantum lasts 8 × (BRP+1) clock cycles.
TSEG2	[20:16]	rw	**Time Segment After Sample Point** (TSEG2 + 1) time quanta is the user-defined nominal time between the sample point and the start of the next synchronization segment. It may be shortened due to re-synchronization. Valid values for TSEG2 are 1 to 31.

Field	Bits	Type	Description
TSEG1	[27:22]	rw	**Time Segment Before Sample Point** (TSEG1 + 1) time quanta is the user-defined nominal time between the end of the synchronization segment and the sample point. It includes the propagation segment, which takes into account signal propagation delays. The time segment may be lengthened due to re-synchronization. Valid values for TSEG1 are 2 to 63.
0	[7:6], [14:12], 21, [31:28]	r	**Reserved** Read as 0; should be written with 0.

The Fast Node Bit Timing Register, FNBTRx contains all parameters to set up CAN bit timing for Data Bit Rate. FNBTRx can be written only if bit NCRx.CCE is set.

CAN_FNBTRx (x = 0-2)
Fast Node x Bit Timing Register (238_H+x*100_H) **Reset Value: 0000 0000_H**
CAN1_FNBTRy (y = 0-2)
Fast Node y Bit Timing Register (238_H+y*100_H) **Reset Value: 0000 0000_H**

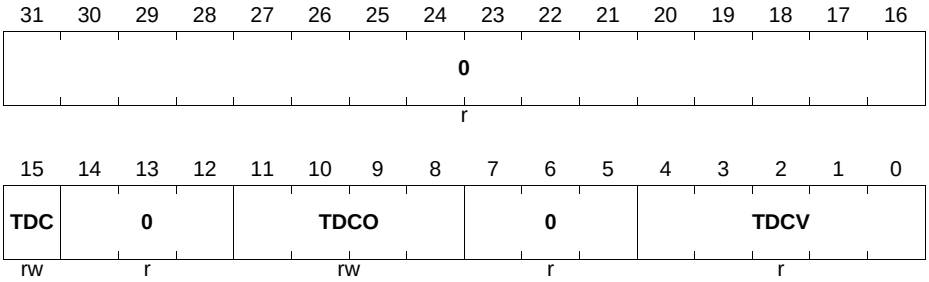


Field	Bits	Type	Description
FBRP	[5:0]	rw	**Fast Baud Rate Prescaler** The duration of one time quantum is given by (BRP + 1) clock cycles.
FSJW	[7:6]	rw	**Fast (Re) Synchronization Jump Width** (SJW + 1) time quanta are allowed for re-synchronization.
FTSEG1	[11:8]	rw	**Fast Time Segment Before Sample Point** (TSEG1 + 1) time quanta is the user-defined nominal time between the end of the synchronization segment and the sample point. It includes the propagation segment, which takes into account signal propagation delays. The time segment may be lengthened due to re-synchronization.
FTSEG2	[14:12]	rw	**Fast Time Segment After Sample Point** (TSEG2 + 1) time quanta is the user-defined nominal time between the sample point and the start of the next synchronization segment. It may be shortened due to re-synchronization.
0	[31:15]	r	**Reserved** Read as 0; should be written with 0.

The Transmitter Delay Compensation Register, TDCRx contains all parameters to setup the Transmitter Delay Compensation Feature and the corresponding status bits. NTDCRx register can be written only if bit NCRx.CCE is set.

CAN_NTDCRx (x = 0-2)
Node x Transmitter Delay Compensation Register
($23C_H$+x*100_H) **Reset Value: 0000 0000_H**

CAN1_NTDCRz (z = 0-2)
Node z Transmitter Delay Compensation Register
($23C_H$+z*100_H) **Reset Value: 0000 0000_H**

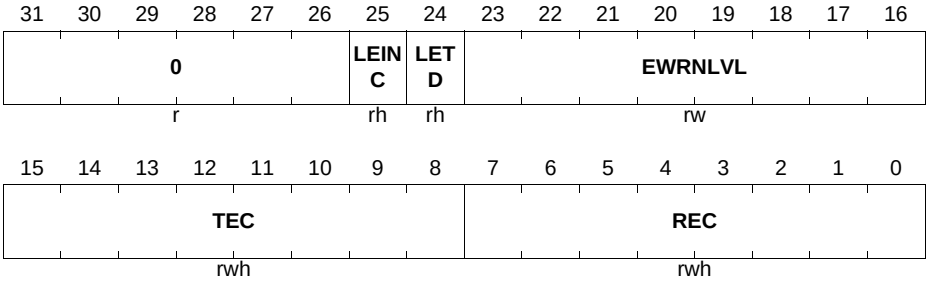


Field	Bits	Type	Description
TDCV	[4:0]	r	**Transmitter Delay Compensation Value** This bit field shows the secondary sample point which is the sum of the measured Transmitter Delay (from CAN Transmit to Receive) and Transmitter Delay compensation offset, NTDCRz.TDCO. Valid values for TDCV are 0 to 31 times of time quanta t_Q.
TDCO	[11:8]	rw	**Transmitter Delay Compensation Offset** This bit field defines the Transmitter Delay compensation offset that is added to the measured Transmitter Delay (from CAN Transmit to Receive) which forms the secondary sample point. Valid values for TDCO are 0 to 15 times of time quanta t_Q.

Field	Bits	Type	Description
TDC	15	rw	**Transmitter Delay Compensation Enable** This bit enables the Transmitter Delay Compensation feature: 0_B Transmitter Delay Compensation disabled. 1_B Transmitter Delay Compensation enabled.
0	[7:5], [14:12], [31:16]	r	**Reserved** Read as 0; should be written with 0.

The Node Error Counter Register NECNTx contains the CAN receive and transmit error counter as well as some additional bits to ease error analysis. NECNTx can be written only if bit NCRx.CCE is set.

CAN_NECNTx (x = 0-2)
Node x Error Counter Register **(214_H+x*100_H)** **Reset Value: 0060 0000$_H$**
CAN1_**NECNTy (y=0-2)**
Node y Error Counter Register **(214_H+y*100_H)** **Reset Value: 0060 0000$_H$**

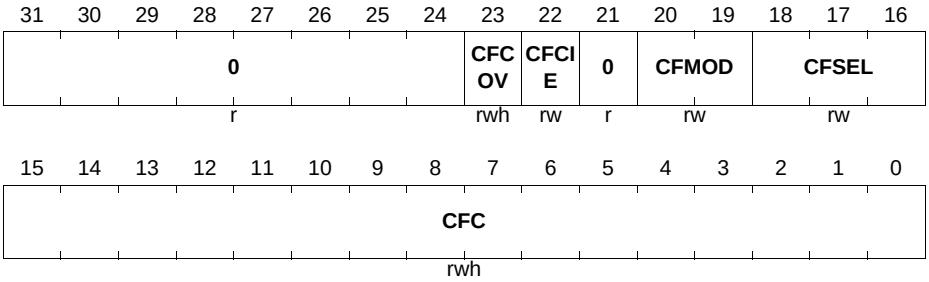


Field	Bits	Type	Description
REC	[7:0]	rwh	**Receive Error Counter** Bit field REC contains the value of the receive error counter of CAN node x.
TEC	[15:8]	rwh	**Transmit Error Counter** Bit field TEC contains the value of the transmit error counter of CAN node x.
EWRNLVL	[23:16]	rw	**Error Warning Level** Bit field EWRNLVL determines the threshold value (warning level, default 96) to be reached in order to set the corresponding error warning bit EWRN.
LETD	24	rh	**Last Error Transfer Direction** 0_B The last error occurred while the CAN node x was receiver (REC has been incremented). 1_B The last error occurred while the CAN node x was transmitter (TEC has been incremented).
LEINC	25	rh	**Last Error Increment** 0_B The last error led to an error counter increment of 1. 1_B The last error led to an error counter increment of 8.

Field	Bits	Type	Description
0	[31:26]	r	**Reserved** Read as 0; should be written with 0.

The Node Frame Counter Register NFCRx contains the actual value of the frame counter as well as control and status bits of the frame counter.

CAN_NFCRx (x = 0-2)
Node x Frame Counter Register ($218_H+x*100_H$) **Reset Value: 0000 0000$_H$**
CAN1_**NFCRy (y=0-2)**
Node y Frame Counter Register ($218_H+y*100_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CFC	[15:0]	rwh	**CAN Frame Counter** In Frame Count Mode (CFMOD = 00_B), this bit field contains the frame count value. In Time Stamp Mode (CFMOD = 01_B), this bit field contains the captured bit time count value, captured with the start of a new frame. In all Bit Timing Analysis Modes[1] (CFMOD = 10_B), CFC always displays the number of f_{CAN} clock cycles (measurement result) minus 1. Example: a CFC value of 34 in measurement mode CFSEL = 000_B means that 35 f_{CAN} clock cycles have been elapsed between the most recent two dominant edges on the receive input. In Error Count Mode (CFMOD = 11_B), this bit field contains the total amount of error frames received or error detected by the node.

Field	Bits	Type	Description
CFSEL	[18:16]	rw	**CAN Frame Count Selection** This bit field selects the function of the frame counter for the chosen frame count mode. **Frame Count Mode** Bit 0 If Bit 0 of CFSEL is set, then CFC is incremented each time a foreign frame (i.e. a frame not matching to a message object) has been received on the CAN bus. Bit 1 If Bit 1 of CFSEL is set, then CFC is incremented each time a frame matching to a message object has been received on the CAN bus. Bit 2 If Bit 2 of CFSEL is set, then CFC is incremented each time a frame has been transmitted successfully by the node. **Time Stamp Mode** The frame counter is incremented (internally) at the beginning of a new bit time. The value is sampled during the SOF bit of a new frame. The sampled value is visible in the CFC field. **Bit Timing Mode** The available bit timing measurement modes are shown in **Table 21-13**. If CFCIE is set, then an interrupt on request node x (where x is the CAN node number) is generated with a CFC update. **Error Count Mode** The frame counter is incremented when an error frame is received or an error is detected by the node. (001_B to 110_B) (see **Table 21-12** for **Encoding of the LEC Bit field**).
CFMOD	[20:19]	rw	**CAN Frame Counter Mode** This bit field determines the operation mode of the frame counter. 00_B Frame Count Mode: The frame counter is incremented upon the reception and transmission of frames. 01_B Time Stamp Mode: The frame counter is used to count bit times. 10_B Bit Timing Mode: The frame counter is used for analysis of the bit timing. 11_B Error Count Mode: The frame counter is used for counting when an error frame is received or an error is detected by the node.

Field	Bits	Type	Description
CFCIE	22	rw	**CAN Frame Count Interrupt Enable** CFCIE enables the CAN frame counter overflow interrupt of CAN node x. 0_B CAN frame counter overflow interrupt is disabled. 1_B CAN frame counter overflow interrupt is enabled. Bit field NIPRx.CFCINP selects the interrupt output line that is activated at this type of interrupt.
CFCOV	23	rwh	**CAN Frame Counter Overflow Flag** Flag CFCOV is set upon a frame counter overflow (transition from $FFFF_H$ to 0000_H). In bit timing analysis mode, CFCOV is set upon an update of CFC. An interrupt request is generated if CFCIE = 1. 0_B No overflow has occurred since last flag reset. 1_B An overflow has occurred since last flag reset. CFCOV must be reset by software.
0	21, [31:24]	r	**Reserved** Read as 0; should be written with 0.

1) The value of NFCRx.CFC is valid one module cycle later when NFCRx.CFCOV is set.

Bit Timing Analysis Modes

Table 21-13 Bit Timing Analysis Modes (CFMOD = 10)

CFSEL	Measurement
000_B	Whenever a dominant edge (transition from 1 to 0) is monitored on the receive input, the time (measured in clock cycles) between this edge and the most recent dominant edge is stored in CFC.
001_B	Whenever a recessive edge (transition from 0 to 1) is monitored on the receive input the time (measured in clock cycles) between this edge and the most recent dominant edge is stored in CFC.
010_B	Whenever a dominant edge is received as a result of a transmitted dominant edge, the time (clock cycles) between both edges is stored in CFC.
011_B	Whenever a recessive edge is received as a result of a transmitted recessive edge, the time (clock cycles) between both edges is stored in CFC.
100_B	Whenever a dominant edge that qualifies for synchronization is monitored on the receive input, the time (measured in clock cycles) between this edge and the most recent sample point is stored in CFC.

Table 21-13 Bit Timing Analysis Modes (CFMOD = 10) (cont'd)

CFSEL	Measurement
101_B	With each sample point, the time (measured in clock cycles) between the start of the new bit time and the start of the previous bit time is stored in CFC[11:0]. Additional information is written to CFC[15:12] at each sample point: CFC[15]: Transmit value of actual bit time CFC[14]: Receive sample value of actual bit time CFC[13:12]: CAN bus information (see **Table 21-14**)
110_B	Reserved, do not use this combination.
111_B	Reserved, do not use this combination.

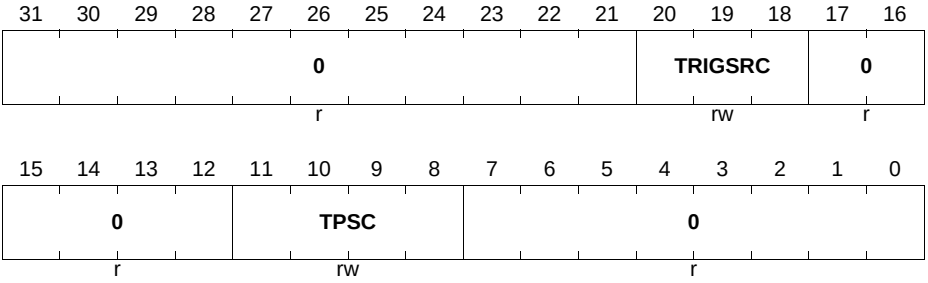
Table 21-14 CAN Bus State Information

CFC[13:12]	CAN Bus State
00_B	**NoBit** The CAN bus is idle, performs bit (de-) stuffing or is in one of the following frame segments: SOF, SRR, CRC, delimiters, first 6 EOF bits, IFS.
01_B	**NewBit** This code represents the first bit of a new frame segment. The current bit is the first bit in one of the following frame segments: Bit 10 (MSB) of standard ID (transmit only), RTR, reserved bits, IDE, DLC(MSB), bit 7 (MSB) in each data byte and the first bit of the ID extension.
10_B	**Bit** This code represents a bit inside a frame segment with a length of more than one bit (not the first bit of those frame segments that is indicated by NewBit). The current bit is processed within one of the following frame segments: ID bits (except first bit of standard ID for transmission and first bit of ID extension), DLC (3 LSB) and bits 6-0 in each data byte.
11_B	**Done** The current bit is in one of the following frame segments: Acknowledge slot, last bit of EOF, active/passive-error frame, overload frame. Two or more directly consecutive Done codes signal an Error Frame.

The Node x Timer Clock Control, Node x Timer Receive Timeout and Node x Timer A/B/C Transmit Trigger Registers offer additional timing functions for the node.

The Node x Timer Clock Control Register NTCCRx controls the functions of the node timer.

CAN_NTCCRx (x = 0-2)
Node x Timer Clock Control Register ($21C_H$+x*100_H) **Reset Value: 0000 0000$_H$**
CAN1_**NTCCRy (y=0-2)**
Node y Timer Clock Control Register ($21C_H$+y*100_H) **Reset Value: 0000 0000$_H$**

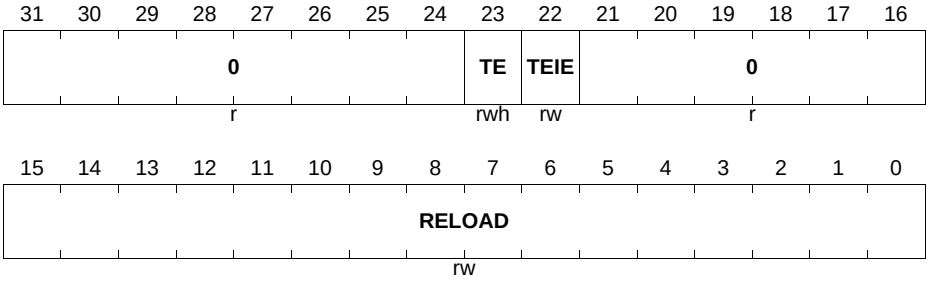


Field	Bits	Type	Description
0	[7:0]	r	**Reserved** Read as 0; should be written with 0.
TPSC	[11:8]	rw	**Timer Prescaler** The duration of one timer clock is given by (TPSC + 1) CAN bit times for all NTCCRx.TRIGSRC settings.
0	[17:12]	r	**Reserved** Read as 0; should be written with 0.

Field	Bits	Type	Description
TRIGSRC	[20:18]	rw	**Trigger Source** This bit selects the trigger source for the different modes in the node timer. 000_B Node x Timer is decremented per f_{CLC} timing to 0. 001_B System Timer (STM) trigger event enabled. Node x Timer is decremented per STM trigger event prescaled by (TPSC + 1). 010_B General Timer (GTM) trigger event enabled. Node x Timer is decremented per GTM trigger event prescaled by (TPSC + 1). 011_B Reserved, do not use. $\ldots_B$ ... 111_B Reserved, do not use.
0	[31:21]	r	**Reserved** Read as 0; should be written with 0.

The Node x Timer Receive Timeout Register NTRTx controls the node timing functions for Receive Timeout Mode.

CAN_NTRTRx (x = 0-2)
Node x Timer Receive Timeout Register (220_H+x*100_H) **Reset Value: 0000 0000_H**
CAN1_**NTRTRy (y=0-2)**
Node y Timer Receive Timeout Register (220_H+y*100_H) **Reset Value: 0000 0000_H**

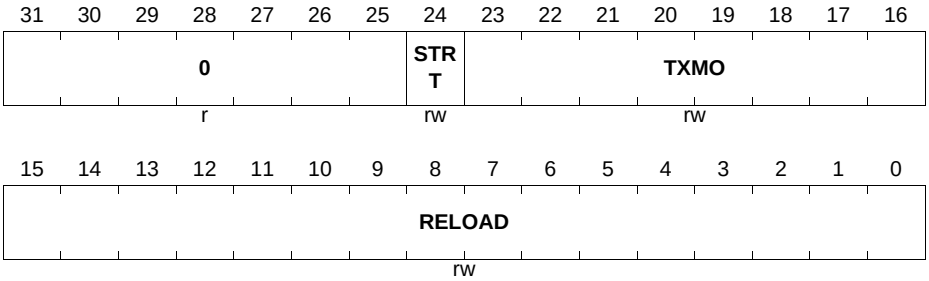


Field	Bits	Type	Description
RELOAD	[15:0]	rw	**Reload Value** This bit field contains the reload value for the timer. The timer will restart when RELOAD is written.
0	[21:16]	r	**Reserved** Read as 0; should be written with 0.
TEIE	22	rw	**Timer Event Interrupt Enable** This bit enables the node timer event interrupt of CAN node x. 0_B Timer event interrupt is disabled 1_B Timer event interrupt is enabled Bit field NIPRx.TEINP selects the interrupt output line which becomes activated at this type of interrupt.
TE	23	rwh	**Timer Event** This flag is set on a node timer transition from 1 to 0 in Receive Timeout Mode. This bit must be reset (i.e Write to '0') by software, writing a '1' has no effect. 0_B No timer event has occurred since last flag reset 1_B Timer event has occurred since last flag reset An interrupt request is generated if TEIE = 1.

Field	Bits	Type	Description
0	[31:24]	r	**Reserved** Read as 0; should be written with 0.

The Node x Timer A Transmit Trigger Register NTATTx controls the node timing functions for Transmit Trigger Mode.

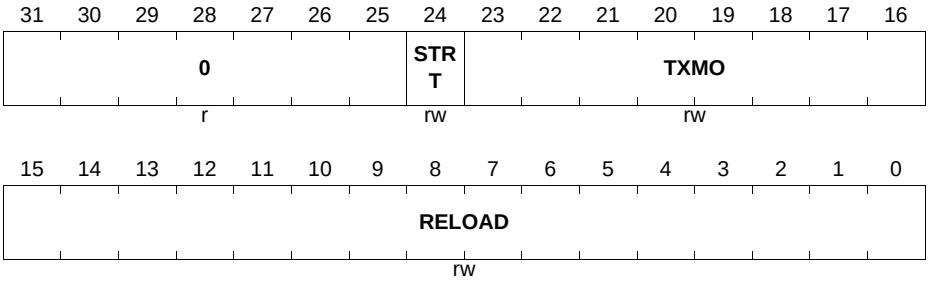
CAN_NTATTRx (x = 0-2)
Node x Timer A Transmit Trigger Register (224_H+x*100_H) Reset Value: 0000 0000_H
CAN1_**NTATTRy (y=0-2)**
Node y Timer A Transmit Trigger Register (224_H+y*100_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
RELOAD	[15:0]	rw	**Reload Value** This bit field contains the reload value for the timer. The timer will restart when RELOAD is written.
TXMO	[23:16]	rw	**Transmit Message Object** On a transmit trigger event selected by bit TRIGSRC, the transmit request bit MOSTATn.TXRQ of message object number n will be set. Note: The NTATTRx.STRT timer has to be stopped before a new value can be programmed into TXMO, i.e 1) Program NTATTR.STRT=0 2) Program NTATTR.TXMO with new value
STRT	24	rw	**Timer Start** This bit field controls the operation of the timer. 0_B Timer is stopped. 1_B Timer is started.
0	[31:25]	r	**Reserved** Read as 0; should be written with 0.

The Node x Timer B Transmit Trigger Register NTBTTx controls the node timing functions for Transmit Trigger Mode.

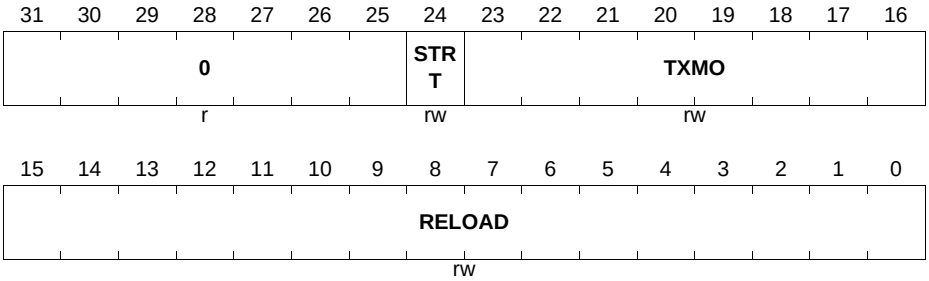
CAN_NTBTTRx (x = 0-2)
Node x Timer B Transmit Trigger Register (228_H+x*100_H) Reset Value: 0000 0000_H
CAN1_**NTBTTRy (y=0-2)**
Node y Timer B Transmit Trigger Register (228_H+y*100_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
RELOAD	[15:0]	rw	**Reload Value** This bit field contains the reload value for the timer. The timer will restart when RELOAD is written.
TXMO	[23:16]	rw	**Transmit Message Object** On a transmit trigger event selected by bit TRIGSRC, the transmit request bit MOSTATn.TXRQ of message object number n will be set. Note: The NTBTTRx.STRT timer has to be stopped before a new value can be programmed into TXMO, i.e 1) Program NTBTTR.STRT=0 2) Program NTBTTR.TXMO with new value
STRT	24	rw	**Timer Start** This bit field controls the operation of the timer. 0_B Timer is stopped. 1_B Timer is started.
0	[31:25]	r	**Reserved** Read as 0; should be written with 0.

The Node x Timer C Transmit Trigger Register NTCTTx controls the node timing functions for Transmit Trigger Mode.

CAN_NTCTTRx (x = 0-2)
Node x Timer C Transmit Trigger Register ($22C_H$+x*100_H) Reset Value: 0000 0000_H
CAN1_**NTCTTRy (y=0-2)**
Node y Timer C Transmit Trigger Register ($22C_H$+y*100_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
RELOAD	[15:0]	rw	**Reload Value** This bit field contains the reload value for the timer. The timer will restart when RELOAD is written.
TXMO	[23:16]	rw	**Transmit Message Object** On a transmit trigger event selected by bit TRIGSRC, the transmit request bit MOSTATn.TXRQ of message object number n will be set. Note: The NTCTTRx.STRT timer has to be stopped before a new value can be programmed into TXMO, i.e 1) Program NTCTTR.STRT=0 2) Program NTCTTR.TXMO with new value
STRT	24	rw	**Timer Start** This bit field controls the operation of the timer. 0_B Timer is stopped. 1_B Timer is started.
0	[31:25]	r	**Reserved** Read as 0; should be written with 0.

21.6.3 Message Object Registers

The Message Object Control Register MOCTRn and the Message Object Status Register MOSTATn are located at the same address offset within a message object address block (offset address $1C_H$). The MOCTRn is a write-only register that makes it possible to set/reset CAN transfer related control bits through software. Therefore the reset value is written as 0x0, even though the read part of the register has a different reset value.

CAN_MOCTRz (z = 0-126)
Message Object z Control Register($101C_H+z*20_H$) **Reset Value: 00000000_H**
CAN1_**MOCTRw (w=0-126)**
Message Object w Control Register($101C_H+w*20_H$) **Reset Value: 0000 0000_H**
CAN_MOCTR127
Message Object 127 Control Register($1FFC_H$) **Reset Value: 00000000_H**
CAN1_MOCTR127
Message Object 127 Control Register($1FFC_H$) **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				SET DIR	SET TXEN1	SET TXEN0	SET TXRQ	SET RXEN	SET RTSEL	SET MSGVAL	SET MSGLST	SET NEWDAT	SET RXUPD	SET TXPND	SET RXPND
w				w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				RES DIR	RES TXEN1	RES TXEN0	RES TXRQ	RES RXEN	RES RTSEL	RES MSGVAL	RES MSGLST	RES NEWDAT	RES RXUPD	RES TXPND	RES RXPND
w				w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
RESRXPND, SETRXPND	0, 16	w	**Reset/Set Receive Pending** These bits control the set/reset condition for RXPND (see **Table 21-15**).
RESTXPND, SETTXPND	1, 17	w	**Reset/Set Transmit Pending** These bits control the set/reset condition for TXPND (see **Table 21-15**).
RESRXUPD, SETRXUPD	2, 18	w	**Reset/Set Receive Updating** These bits control the set/reset condition for RXUPD (see **Table 21-15**).

Field	Bits	Type	Description
RESNEWDAT, SETNEWDAT	3, 19	w	**Reset/Set New Data** These bits control the set/reset condition for NEWDAT (see **Table 21-15**).
RESMSGLST, SETMSGLST	4, 20	w	**Reset/Set Message Lost** These bits control the set/reset condition for MSGLST (see **Table 21-15**).
RESMSGVAL, SETMSGVAL	5, 21	w	**Reset/Set Message Valid** These bits control the set/reset condition for MSGVAL (see **Table 21-15**).
RESRTSEL, SETRTSEL	6, 22	w	**Reset/Set Receive/Transmit Selected** These bits control the set/reset condition for RTSEL (see **Table 21-15**).
RESRXEN, SETRXEN	7, 23	w	**Reset/Set Receive Enable** These bits control the set/reset condition for RXEN (see **Table 21-15**).
RESTXRQ, SETTXRQ	8, 24	w	**Reset/Set Transmit Request** These bits control the set/reset condition for TXRQ (see **Table 21-15**).
RESTXEN0, SETTXEN0	9, 25	w	**Reset/Set Transmit Enable 0** These bits control the set/reset condition for TXEN0 (see **Table 21-15**).
RESTXEN1, SETTXEN1	10, 26	w	**Reset/Set Transmit Enable 1** These bits control the set/reset condition for TXEN1 (see **Table 21-15**).
RESDIR, SETDIR	11, 27	w	**Reset/Set Message Direction** These bits control the set/reset condition for DIR (see **Table 21-15**).
0	[15:12], [31:28]	w	**Reserved** Should be written with 0.

Table 21-15 Reset/Set Conditions for Bits in Register MOCTRn

RESy Bit[1)]	SETy Bit	Action on Write
Write 0	Write 0	Leave element unchanged
	No write	
No write	Write 0	
Write 1	Write 1	
Write 1	Write 0	Reset element
	No write	
Write 0	Write 1	Set element
No write		

1) The parameter “y” stands for the second part of the bit name (“RXPND”, “TXPND”, … up to “DIR”).

The MOSTATn is a read-only register that indicates message object list status information such as the number of the current message object predecessor and successor message object, as well as the list number to which the message object is assigned.

CAN_MOSTAT0
Message Object 0 Status Register ($101C_H$) Reset Value: 0100 0000_H
CAN1_**MOSTAT0**
Message Object 0 Status Register ($101C_H$) Reset Value: 0100 0000_H
CAN_MOSTATn (n = 1-126)
Message Object n Status Register($101C_H$+n*20_H)
Reset Value: ((n+1)*01000000_H)+((n-1)*00010000_H)
CAN1_**MOSTATw (w = 1-126)**
Message Object w Status Register($101C_H$+w*20_H)
Reset Value: ((w+1)*01000000_H)+((w-1)*00010000_H)
CAN_MOSTAT127
Message Object 127 Status Register ($1FFC_H$) Reset Value: 7F7E 0000_H
CAN1_**MOSTAT127**
Message Object 127 Status Register ($1FFC_H$) Reset Value: 7F7E 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PNEXT								PPREV							
rh								rh							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LIST				DIR	TX EN1	TX EN0	TX RQ	RX EN	RTS EL	MSG VAL	MSG LST	NEW DAT	RX UPD	TX PND	RX PND
rh				rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
RXPND	0	rh	**Receive Pending** 0_B No CAN message has been received. 1_B A CAN message has been received by the message object n, either directly or via gateway copy action. RXPND is set by hardware and must be reset by software.

Field	Bits	Type	Description
TXPND	1	rh	**Transmit Pending** 0_B No CAN message has been transmitted. 1_B A CAN message from message object n has been transmitted successfully over the CAN bus. TXPND is set by hardware and must be reset by software.
RXUPD	2	rh	**Receive Updating** 0_B No receive update ongoing. 1_B Message identifier, DLC, and data of the message object are currently updated.
NEWDAT	3	rh	**New Data** 0_B No update of the message object n since last flag reset. 1_B Message object n has been updated. NEWDAT is set by hardware after a received CAN frame has been stored in message object n. NEWDAT is cleared by hardware when a CAN transmission of message object n has been started. NEWDAT should be set by software after the new transmit data has been stored in message object n to prevent the automatic reset of TXRQ at the end of an ongoing transmission.
MSGLST	4	rh	**Message Lost** 0_B No CAN message is lost. 1_B A CAN message is lost because NEWDAT has become set again when it has already been set.
MSGVAL	5	rh	**Message Valid** 0_B Message object n is not valid. 1_B Message object n is valid. Only a valid message object takes part in CAN transfers.

Field	Bits	Type	Description
RTSEL	6	rh	**Receive/Transmit Selected** 0_B Message object n is not selected for receive or transmit operation. 1_B Message object n is selected for receive or transmit operation. **Frame Reception:** RTSEL is set by hardware when message object n has been identified for storage of a CAN frame that is currently received. Before a received frame becomes finally stored in message object n, a check is performed to determine if RTSEL is set. Thus the CPU can suppress a scheduled frame delivery to this message object n by clearing RTSEL by software. **Frame Transmission:** RTSEL is set by hardware when message object n has been identified to be transmitted next. A check is performed to determine if RTSEL is still set before message object n is actually set up for transmission and bit NEWDAT is cleared. It is also checked that RTSEL is still set before its message object n is verified due to the successful transmission of a frame. RTSEL needs to be checked only when the context of message object n changes, and a conflict with an ongoing frame transfer shall be avoided. In all other cases, RTSEL can be ignored. RTSEL has no impact on message acceptance filtering. RTSEL is not cleared by hardware.
RXEN	7	rh	**Receive Enable** 0_B Message object n is not enabled for frame reception. 1_B Message object n is enabled for frame reception. RXEN is evaluated for receive acceptance filtering only.

Field	Bits	Type	Description
TXRQ	8	rh	**Transmit Request** 0_B No transmission of message object n is requested. 1_B Transmission of message object n on the CAN bus is requested. The transmit request becomes valid only if TXRQ, TXEN0, TXEN1 and MSGVAL are set. TXRQ is set by hardware if a matching Remote Frame has been received correctly. TXRQ is reset by hardware if message object n has been transmitted successfully and NEWDAT is not set again by software.
TXEN0	9	rh	**Transmit Enable 0** 0_B Message object n is not enabled for frame transmission. 1_B Message object n is enabled for frame transmission. Message object n can be transmitted only if both bits, TXEN0 and TXEN1, are set. The user may clear TXEN0 in order to inhibit the transmission of a message that is currently updated, or to disable automatic response of Remote Frames.
TXEN1	10	rh	**Transmit Enable 1** 0_B Message object n is not enabled for frame transmission. 1_B Message object n is enabled for frame transmission. Message object n can be transmitted only if both bits, TXEN0 and TXEN1, are set. TXEN1 is used by the MultiCAN+ module for selecting the active message object in the Transmit FIFOs.

Field	Bits	Type	Description
DIR	11	rh	**Message Direction** 0_B Receive Object selected: With TXRQ = 1, a Remote Frame with the identifier of message object n is scheduled for transmission. On reception of a Data Frame with matching identifier, the message is stored in message object n. 1_B Transmit Object selected: If TXRQ = 1, message object n is scheduled for transmission of a Data Frame. On reception of a Remote Frame with matching identifier, bit TXRQ is set.
LIST	[15:12]	rh	**List Allocation** LIST indicates the number of the message list to which message object n is allocated. LIST is updated by hardware when the list allocation of the object is modified by a panel command.
PPREV	[23:16]	rh	**Pointer to Previous Message Object** PPREV holds the message object number of the previous message object in a message list structure.
PNEXT	[31:24]	rh	**Pointer to Next Message Object** PNEXT holds the message object number of the next message object in a message list structure.

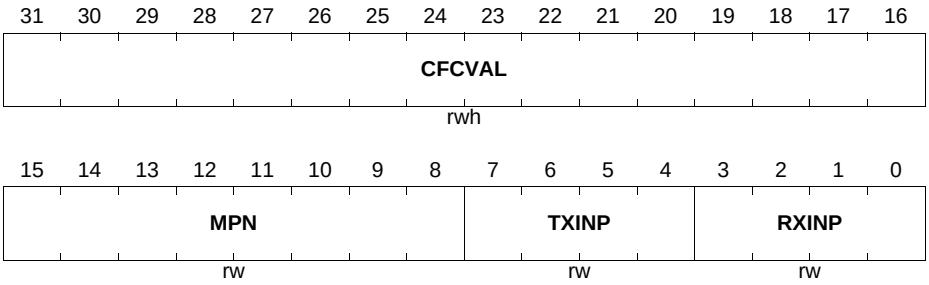
Table 21-16 MOSTATn Reset Values

Message Object	PNEXT	PPREV	Reset Value
0	1	0	$0100\ 0000_H$
1	2	0	$0200\ 0000_H$
2	3	1	$0301\ 0000_H$
3	4	2	$0402\ 0000_H$
…	…	…	…
127	127	126	$7F7E\ 0000_H$

The Message Object Interrupt Pointer Register MOIPRn holds the message interrupt pointers, the message pending number, and the frame counter value of message object n.

CAN_MOIPRn (n = 0-127)
Message Object n Interrupt Pointer Register
(1008_H+n*20_H) **Reset Value: 0000 0000_H**

CAN1_**MOIPRw (w = 0 - 127)**
Message Object w Interrupt Pointer Register
(1008_H+w*20_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
RXINP	[3:0]	rw	**Receive Interrupt Node Pointer** RXINP selects the interrupt output line INT_Om (m = 0-15), CAN and (m = 0-7), CAN1 for a receive interrupt event of message object n. RXINP can also be taken for message pending bit selection (see **Page 21-48**). Option 1000_B to 1111_B is Reserved and not applicable for CAN1**.** 0000_B Interrupt output line INT_O0 is selected. 0001_B Interrupt output line INT_O1 is selected. $\cdots_B$ … 1110_B Interrupt output line INT_O14 is selected. 1111_B Interrupt output line INT_O15 is selected.

Field	Bits	Type	Description
TXINP	[7:4]	rw	**Transmit Interrupt Node Pointer** TXINP selects the interrupt output line INT_Om (m = 0-15), CAN and (m = 0-7), CAN1 for a transmit interrupt event of message object n. TXINP can also be taken for message pending bit selection (see **Page 21-48**). Option 1000_B to 1111_B is Reserved and not applicable for CAN1**.** 0000_B Interrupt output line INT_O0 is selected. 0001_B Interrupt output line INT_O1 is selected. $\ldots_B$ … 1110_B Interrupt output line INT_O14 is selected. 1111_B Interrupt output line INT_O15 is selected.
MPN	[15:8]	rw	**Message Pending Number** This bit field selects the bit position of the bit in the Message Pending Register that is set upon a message object n receive/transmit interrupt.
CFCVAL	[31:16]	rwh	**CAN Frame Counter Value** When a message is stored in message object n or message object n has been successfully transmitted, the CAN frame counter value NFCRx.CFC is then copied to CFCVAL.

The Message Object Function Control Register MOFCRn contains bits that select and configure the function of the message object. It also holds the CAN data length code.

CAN_MOFCRn (n = 0-127)
Message Object n Function Control Register
(1000_H+n*20_H) **Reset Value: 0000 0000_H**

CAN1_**MOFCRw (w = 0 - 127)**
Message Object w Function Control Register
(1000_H+w*20_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				DLC				STT	SDT	RMM	FRR EN	0	OVIE	TXIE	RXIE
rw				rwh				rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				DAT C	DLC C	IDC	GDF S	0	FDF	BRS	RXT OE	MMC			
rw				rw	rw	rw	rw	rw	rwh	rwh	rw	rw			

Field	Bits	Type	Description
MMC	[3:0]	rw	**Message Mode Control** MMC controls the message mode of message object n. 0000_B Standard Message Object 0001_B Receive FIFO Base Object 0010_B Transmit FIFO Base Object 0011_B Transmit FIFO Slave Object 0100_B Gateway Source Object 0101_B CANFD 64 bytes Message Mode $\ldots_B$ ... 1111_B Do not use

Field	Bits	Type	Description
RXTOE	4	rw	**Receive Time-Out Enable** RXTOE enables participation of message in receive time-out check which is performed via a Timer controlled by the register NTRTR of the corresponding node. 0_B Message does not take part in receive time-out check 1_B Message takes part in receive time-out check Applicable to receive messages and to transmit messages waiting for remote frames.
BRS	5	rwh	**Bit Rate Switch** 0_B Message Object transmission/reception without bit rate switching. 1_B Message Object transmission/reception with bit rate switching. Please see **Section 21.4.12.4** and **Table 21-18**
FDF	6	rwh	**CAN FD Frame Format** 0_B Message Object transmission/reception in Classical CAN Frame Format. 1_B Message Object transmission/reception in CAN FD Format (new DLC coding and CRC) Please see **Section 21.4.12.4** and **Table 21-18**
GDFS	8	rw	**Gateway Data Frame Send** 0_B TXRQ is unchanged in the destination object. 1_B TXRQ is set in the gateway destination object after the internal transfer from the gateway source to the gateway destination object. Applicable only to a gateway source object; ignored in other nodes.
IDC	9	rw	**Identifier Copy** 0_B The identifier of the gateway source object is not copied. 1_B The identifier of the gateway source object (after storing the received frame in the source) is copied to the gateway destination object. Applicable only to a gateway source object; ignored in other nodes.

Field	Bits	Type	Description
DLCC	10	rw	**Data Length Code Copy** 0_B Data length code is not copied. 1_B Data length code of the gateway source object (after storing the received frame in the source) is copied to the gateway destination object. Applicable only to a gateway source object; ignored in other nodes.
DATC	11	rw	**Data Copy** 0_B Data fields are not copied. 1_B Data fields in registers MODATALn and MODATAHn of the gateway source object (after storing the received frame in the source) are copied to the gateway destination. Applicable only to a gateway source object; ignored in other nodes.
RXIE	16	rw	**Receive Interrupt Enable** RXIE enables the message receive interrupt of message object n. This interrupt is generated after reception of a CAN message (independent of whether the CAN message is received directly or indirectly via a gateway action). 0_B Message receive interrupt is disabled. 1_B Message receive interrupt is enabled. Bit field MOIPRn.RXINP selects the interrupt output line which becomes activated at this type of interrupt.
TXIE	17	rw	**Transmit Interrupt Enable** TXIE enables the message transmit interrupt of message object n. This interrupt is generated after the transmission of a CAN message. 0_B Message transmit interrupt is disabled. 1_B Message transmit interrupt is enabled. Bit field MOIPRn.TXINP selects the interrupt output line which becomes activated at this type of interrupt.

Field	Bits	Type	Description
OVIE	18	rw	**Overflow Interrupt Enable** OVIE enables the FIFO full interrupt of message object n. This interrupt is generated when the pointer to the current message object (CUR) reaches the value of SEL in the FIFO/Gateway Pointer Register. 0_B FIFO full interrupt is disabled. 1_B FIFO full interrupt is enabled. If message object n is a Receive FIFO base object, bit field MOIPRn.TXINP selects the interrupt output line which becomes activated at this type of interrupt. If message object n is a Transmit FIFO base object, bit field MOIPRn.RXINP selects the interrupt output line which becomes activated at this type of interrupt. For all other message object modes, bit OVIE has no effect.
FRREN	20	rw	**Foreign Remote Request Enable** Specifies whether the TXRQ bit is set in message object n or in a foreign message object referenced by the pointer CUR. 0_B TXRQ of message object n is set on reception of a matching Remote Frame. 1_B TXRQ of the message object referenced by the pointer CUR is set on reception of a matching Remote Frame.
RMM	21	rw	**Transmit Object Remote Monitoring** 0_B Remote monitoring is disabled: Identifier, IDE bit, and DLC of message object n remain unchanged upon the reception of a matching Remote Frame. 1_B Remote monitoring is enabled: Identifier, IDE bit, and DLC of a matching Remote Frame are copied to transmit object n in order to monitor incoming Remote Frames. Bit RMM applies only to transmit objects and has no effect on receive objects.

Field	Bits	Type	Description
SDT	22	rw	**Single Data Transfer** If SDT = 1 and message object n is not a FIFO base object, then MSGVAL is reset when this object has taken part in a successful data transfer (receive or transmit). If SDT = 1 and message object n is a FIFO base object, then MSGVAL is reset when the pointer to the current object CUR reaches the value of SEL in the FIFO/Gateway Pointer Register. With SDT = 0, bit MSGVAL is not affected.
STT	23	rw	**Single Transmit Trial** If this bit is set, then TXRQ is cleared on transmission start of message object n. Thus, no transmission retry is performed in case of transmission failure.
DLC	[27:24]	rwh	**Data Length Code** Bit field determines the number of data bytes for message object n. In Classical CAN Format: A value of DLC > 8 results in a data length of 8 data bytes. If a frame with DLC > 8 is received, the received value is stored in the message object. In CAN FD format, valid values for DLC are 0 to 15. See **Table 21-17**
0	7, [15:12], 19, [31:28]	rw	**Reserved** Read as 0 after reset; value last written is read back; should be written with 0.

Coding of DLC in CAN FD

Table 21-17 Coding of DLC in CAN FD

DLC Value	Description
0000_B	0 Data Byte for Message Object n.
$\cdots_B$	...
1000_B	8 Data Byte for Message Object n.
1001_B	12 Data Byte for Message Object n.
1010_B	16 Data Byte for Message Object n.

Table 21-17 Coding of DLC in CAN FD (cont'd)

DLC Value	Description
1011_B	20 Data Byte for Message Object n.
1100_B	24 Data Byte for Message Object n.
1101_B	32 Data Byte for Message Object n.
1110_B	48 Data Byte for Message Object n.
1111_B	64 Data Byte for Message Object n.

CAN FD Transmit And Receive Behavior

Message transmission and reception behavior of CAN data frames is summarized at **Table 21-18** below.

Table 21-18 CAN FD Transmit And Receive Behavior

FDEN	FDF[1)]	BRS[1)]	Transmit Behavior
0_B	0_B	0_B	Classical CAN Frames (ISO 11898-1)
0_B	0_B	1_B	Classical CAN Frames (ISO 11898-1)
0_B	1_B	0_B	Transmission Cancelled[2)] (i.e TXRQ bit cleared)
0_B	1_B	1_B	Transmission Cancelled[2)] (i.e TXRQ bit cleared)
1_B	0_B	0_B	Classical CAN Frames (ISO 11898-1)
1_B	0_B	1_B	Classical CAN Frames (ISO 11898-1)
1_B	1_B	0_B	Long Frame (i.e CAN FD frame with BRS = 0, whole frame transmitted with slow baudrate)
1_B	1_B	1_B	Long + Fast Frame (i.e CAN FD frame with BRS = 1, switching fast baudrate for dataphase)
FDEN	**FDF[3)]**	**BRS[3)]**	**Receive Behavior[4)]**
0_B	$0/1_B$	$0/1_B$	- Classical CAN Frames
1_B	$0/1_B$	$0/1_B$	- Classical CAN Frames - Long Frames - Long + Fast Frames

1) User writes

2) TXRQ in message object is cleared upon transmit setup and transmit setup cancelled. CAN node is not blocked and able to transmit other message objects.

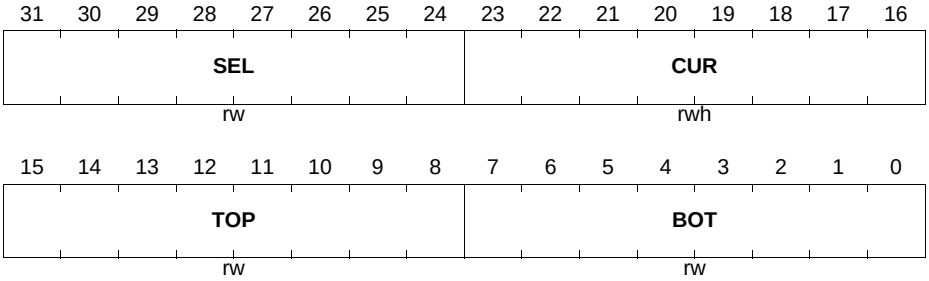
3) Hardware writes

4) Please note that Remote Frames Request do not change BRS and FDF bits.

The Message Object FIFO/Gateway Pointer register MOFGPRn contains a set of message object link pointers that are used for FIFO and gateway operations.

CAN_MOFGPRn (n = 0-127)
Message Object n FIFO/Gateway Pointer Register
(1004_H+n*20_H) **Reset Value: 0000 0000_H**

CAN1_**MOFGPRw (w = 0 - 127)**
Message Object n FIFO/Gateway Pointer Register
(1004_H+w*20_H) **Reset Value: 0000 0000_H**



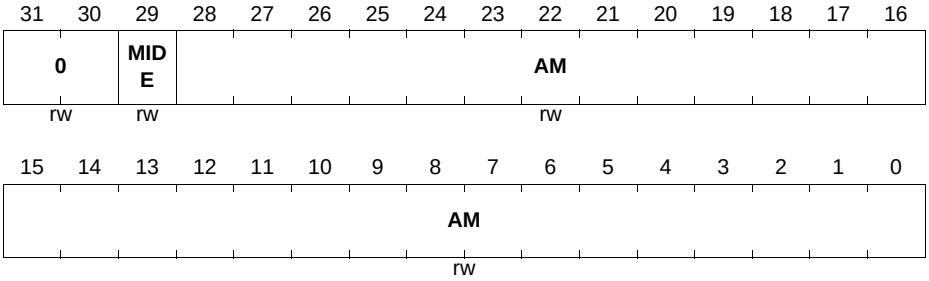
Field	Bits	Type	Description
BOT	[7:0]	rw	**Bottom Pointer** Bit field BOT points to the first element in a FIFO structure. Or in the case when MOFCR.MMC=5, CAN FD 64 bytes message mode, BOT points to where Data bytes 8-35 are stored in the message object.
TOP	[15:8]	rw	**Top Pointer** Bit field TOP points to the last element in a FIFO structure. Or in the case when MOFCR.MMC=5, CAN FD 64 bytes message mode, BOT points to where Data bytes 36-63 are stored in the message object.
CUR	[23:16]	rwh	**Current Object Pointer** Bit field CUR points to the actual target object within a FIFO/Gateway structure. After a FIFO/gateway operation CUR is updated with the message number of the next message object in the list structure (given by PNEXT of the Message Object Status Register) until it reaches the FIFO top element (given by TOP) when it is reset to the bottom element (given by BOT).

Field	Bits	Type	Description
SEL	[31:24]	rw	**Object Select Pointer** Bit field SEL is the second (software) pointer to complement the hardware pointer CUR in the FIFO structure. SEL is used for monitoring purposes (FIFO interrupt generation).

Message Object n Acceptance Mask Register MOAMRn contains the mask bits for the acceptance filtering of the message object n.

CAN_MOAMRn (n = 0-127)
Message Object n Acceptance Mask Register
($100C_H+n*20_H$) **Reset Value: 3FFF FFFF$_H$**

CAN1_**MOAMRw (w = 0 - 127)**
Message Object w Acceptance Mask Register
($100C_H+w*20_H$) **Reset Value: 3FFF FFFF$_H$**

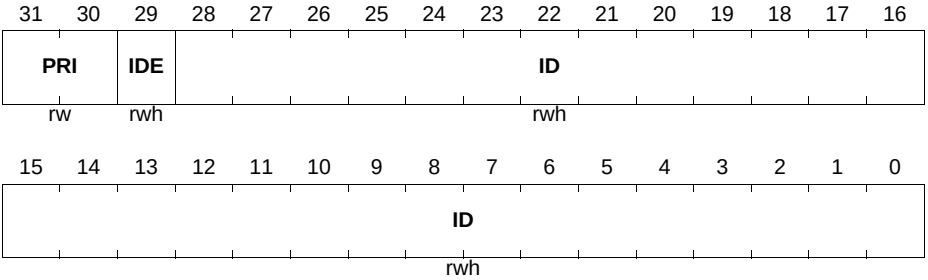


Field	Bits	Type	Description
AM	[28:0]	rw	**Acceptance Mask for Message Identifier** Bit field AM is the 29-bit mask for filtering incoming messages with standard identifiers (AM[28:18]) or extended identifiers (AM[28:0]). For standard identifiers, bits AM[17:0] are “don’t care”.
MIDE	29	rw	**Acceptance Mask Bit for Message IDE Bit** 0_B Message object n accepts the reception of both, standard and extended frames. 1_B Message object n receives frames only with matching IDE bit.
0	[31:30]	rw	**Reserved** Read as 0 after reset; value last written is read back; should be written with 0.

Message Object n Arbitration Register MOARn contains the CAN identifier of the message object.

CAN_MOARn (n = 0-127)
Message Object n Arbitration Register
(1018_H+n*20_H) **Reset Value: 0000 0000_H**

CAN1_**MOARw (w = 0 - 127)**
Message Object w Arbitration Register
(1018_H+w*20_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
ID	[28:0]	rwh	**CAN Identifier of Message Object n** Identifier of a standard message (ID[28:18]) or an extended message (ID[28:0]). For standard identifiers, bits ID[17:0] are “don’t care”.
IDE	29	rwh	**Identifier Extension Bit of Message Object n** 0_B Message object n handles standard frames with 11-bit identifier. 1_B Message object n handles extended frames with 29-bit identifier.

Field	Bits	Type	Description
PRI	[31:30]	rw	**Priority Class** PRI assigns one of the four priority classes 0, 1, 2, 3 to message object n. A lower PRI number defines a higher priority. Message objects with lower PRI value always win acceptance filtering for frame reception and transmission over message objects with higher PRI value. Acceptance filtering based on identifier/mask and list position is performed only between message objects of the same priority class. PRI also determines the acceptance filtering method for transmission: 00_B Reserved. 01_B Transmit acceptance filtering is based on the list order. This means that message object n is considered for transmission only if there is no other message object with valid transmit request (MSGVAL & TXEN0 & TXEN1 = 1) somewhere before this object in the list. 10_B Transmit acceptance filtering is based on the CAN identifier. This means, message object n is considered for transmission only if there is no other message object with higher priority identifier + IDE + DIR (with respect to CAN arbitration rules) somewhere in the list (see **Table 21-19**). 11_B Transmit acceptance filtering is based on the list order (as PRI = 01_B).

Transmit Priority of Msg. Objects based on CAN Arbitration Rules

Table 21-19 Transmit Priority of Msg. Objects Based on CAN Arbitration Rules

Settings of Arbitrarily Chosen Message Objects A and B, (A has higher transmit priority than B)	Comment
A.MOAR[28:18] < B.MOAR[28:18] (11-bit standard identifier of A less than 11-bit standard identifier of B)	Messages with lower standard identifier have higher priority than messages with higher standard identifier. MOAR[28] is the most significant bit (MSB) of the standard identifier. MOAR[18] is the least significant bit of the standard identifier.
A.MOAR[28:18] = B.MOAR[28:18] A.MOAR.IDE = 0 (send Standard Frame) B.MOAR.IDE = 1 (send Extended Frame)	Standard Frames have higher transmit priority than Extended Frames with equal standard identifier.
A.MOAR[28:18] = B.MOAR[28:18] A.MOAR.IDE = B.MOAR.IDE = 0 A.MOSTAT.DIR = 1 (send Data Frame) B.MOSTAT.DIR = 0 (send Remote Fame)	Standard Data Frames have higher transmit priority than standard Remote Frames with equal identifier.
A.MOAR[28:0] = B.MOAR[28:0] A.MOAR.IDE = B.MOAR.IDE = 1 A.MOSTAT.DIR = 1 (send Data Frame) B.MOSTAT.DIR = 0 (send Remote Frame)	Extended Data Frames have higher transmit priority than Extended Remote Frames with equal identifier.
A.MOAR[28:0] < B.MOAR[28:0] A.MOAR.IDE = B.MOAR.IDE = 1 (29-bit identifier)	Extended Frames with lower identifier have higher transmit priority than Extended Frames with higher identifier. MOAR[28] is the most significant bit (MSB) of the overall identifier (standard identifier MOAR[28:18] and identifier extension MOAR[17:0]). MOAR[0] is the least significant bit (LSB) of the overall identifier.

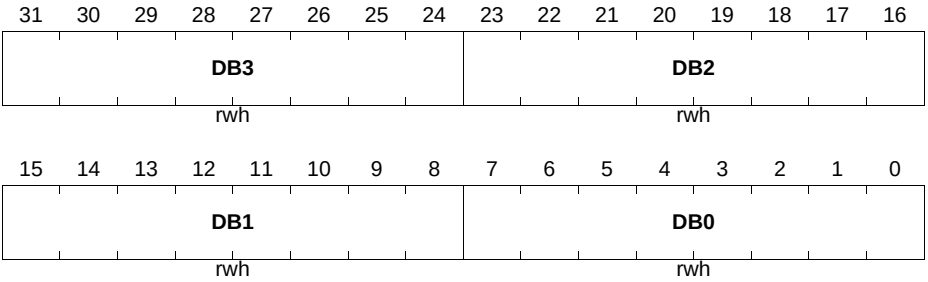
Message Object n Data Register Low MODATALn contains the lowest four data bytes of message object n. Unused data bytes are set to zero upon reception and ignored for transmission.

CAN_MODATALn (n = 0-127)
Message Object n Data Register Low

(1010_H+n*20_H) **Reset Value: 0000 0000_H**

CAN1_**MODATALw (w = 0 - 127)**
Message Object w Data Register Low

(1010_H+w*20_H) **Reset Value: 0000 0000_H**

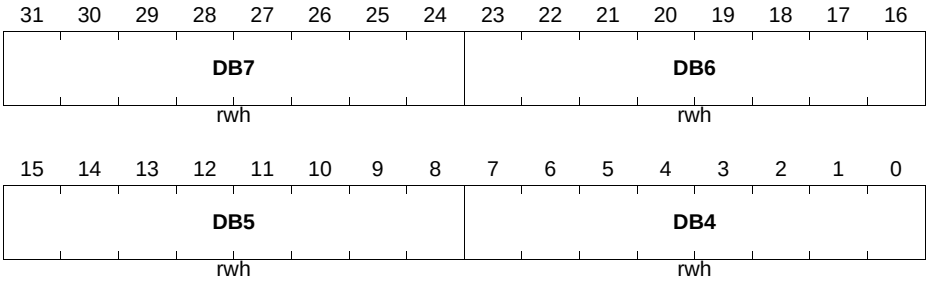


Field	Bits	Type	Description
DB0	[7:0]	rwh	**Data Byte 0 of Message Object n**
DB1	[15:8]	rwh	**Data Byte 1 of Message Object n**
DB2	[23:16]	rwh	**Data Byte 2 of Message Object n**
DB3	[31:24]	rwh	**Data Byte 3 of Message Object n**

Message Object n Data Register High MODATAH contains the highest four data bytes of message object n. Unused data bytes are set to zero upon reception and ignored for transmission.

CAN_MODATAHn (n = 0-127)
Message Object n Data Register High
(1014_H+n*20_H) **Reset Value: 0000 0000_H**

CAN1_**MODATAHw (w = 0 - 127)**
Message Object w Data Register High
(1014_H+w*20_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
DB4	[7:0]	rwh	**Data Byte 4 of Message Object n**
DB5	[15:8]	rwh	**Data Byte 5 of Message Object n**
DB6	[23:16]	rwh	**Data Byte 6 of Message Object n**
DB7	[31:24]	rwh	**Data Byte 7 of Message Object n**

Extended Message Object n Data Register represents the alternate register view for MOFCR, MOFGPR, MOIPR, MOAMR, MODATAL, MODATAH and MOAR registers as pointed using (MOFGPR.TOP, MOFGPR.BOT) on the message object with message mode chosen in CAN FD 64 bytes message mode, (i.e MOFCR.MMC = 5). (See **Section 21.4.12.10**)

CAN_EMOzDATA0 (z = 0-127)
Extended Message Object z Data 0 Register
(1000_H+z*20_H) **Reset Value: 0000 0000_H**

CAN_EMOzDATA1 (z = 0-127)
Extended Message Object z Data 1 Register
(1004_H+z*20_H) **Reset Value: 0000 0000_H**

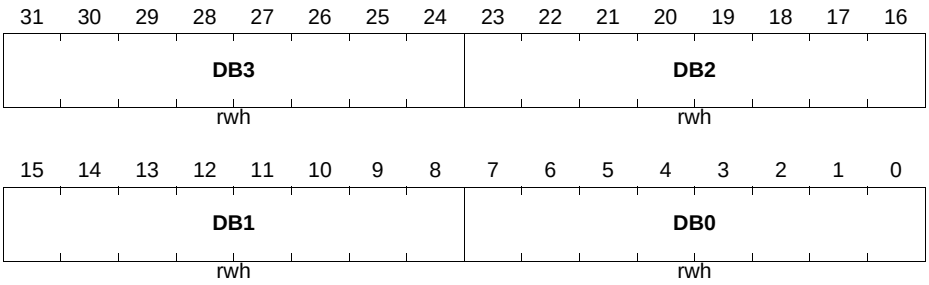
CAN_EMOzDATA2 (z = 0-127)
Extended Message Object z Data 2 Register
(1008_H+z*20_H) **Reset Value: 0000 0000_H**

CAN_EMOzDATA3 (z = 0-127)
Extended Message Object z Data 3 Register
($100C_H$+z*20_H) **Reset Value: 0000 0000_H**

CAN_EMOzDATA4 (z = 0-127)
Extended Message Object z Data 4 Register
(1010_H+z*20_H) **Reset Value: 0000 0000_H**

CAN_EMOzDATA5 (z = 0-127)
Extended Message Object z Data 5 Register
(1014_H+z*20_H) **Reset Value: 0000 0000_H**

CAN_EMOzDATA6 (z = 0-127)
Extended Message Object z Data 6 Register
(1018_H+z*20_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
DB0	[7:0]	rwh	**Data Byte 0 of Message Object n**
DB1	[15:8]	rwh	**Data Byte 1 of Message Object n**

Field	Bits	Type	Description
DB2	[23:16]	rwh	**Data Byte 2 of Message Object n**
DB3	[31:24]	rwh	**Data Byte 3 of Message Object n**

CAN1_EMOwDATA0 (w = 0 - 127)
Extended Message Object w Data 0 Register
(1000_H+w*20_H) **Reset Value: 0000 0000_H**

CAN1_EMOwDATA1 (w = 0 - 127)
Extended Message Object w Data 1 Register
(1004_H+w*20_H) **Reset Value: 0000 0000_H**

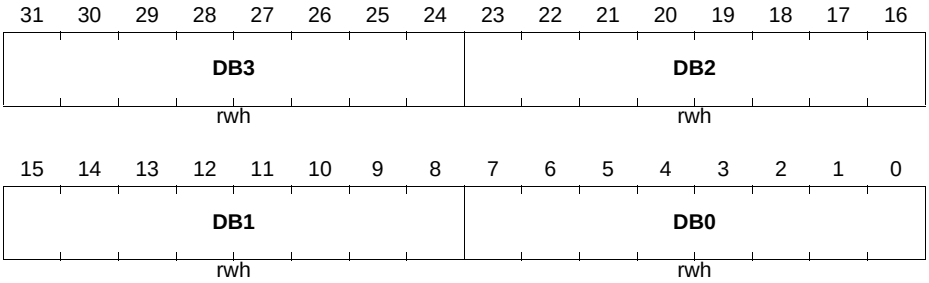
CAN1_EMOwDATA2 (w = 0 - 127)
Extended Message Object w Data 2 Register
(1008_H+w*20_H) **Reset Value: 0000 0000_H**

CAN1_EMOwDATA3 (w = 0 - 127)
Extended Message Object w Data 3 Register
($100C_H$+w*20_H) **Reset Value: 0000 0000_H**

CAN1_EMOwDATA4 (w = 0 - 127)
Extended Message Object w Data 4 Register
(1010_H+w*20_H) **Reset Value: 0000 0000_H**

CAN1_EMOwDATA5 (w = 0 - 127)
Extended Message Object w Data 5 Register
(1014_H+w*20_H) **Reset Value: 0000 0000_H**

CAN1_EMOwDATA6 (w = 0 - 127)
Extended Message Object w Data 6 Register
(1018_H+w*20_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
DB0	[7:0]	rwh	**Data Byte 0 of Message Object n**
DB1	[15:8]	rwh	**Data Byte 1 of Message Object n**

Field	Bits	Type	Description
DB2	[23:16]	rwh	**Data Byte 2 of Message Object n**
DB3	[31:24]	rwh	**Data Byte 3 of Message Object n**

21.7 MultiCAN+ Module Implementation

This section describes CAN module interfaces with the clock control, port connections, interrupt control, and address decoding.

21.7.1 Interfaces of the MultiCAN+ Module

Figure 21-28 shows the TC21x/TC22x/TC23x specific implementation details and interconnections of the MultiCAN+ module. The I/O lines of the MultiCAN+ module (two I/O lines of each CAN node) are connected to the Ports listed in**Table 21-22**. The MultiCAN+ module is also supplied by clock control, interrupt control, and address decoding logic. MultiCAN+ interrupts can be directed to the DMA, CPU, GTM modules which are able to trigger DMA transfers and GTM, CPU operations.

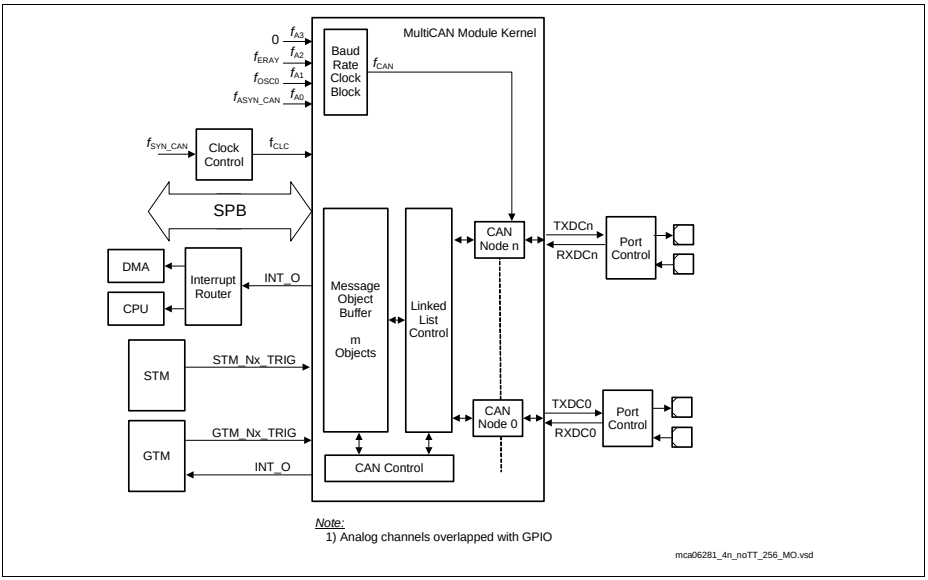


Figure 21-28 MultiCAN+ Module Implementation and Interconnections with n := 3 and m := 128 for AURIX
MultiCAN1 Module Implementation and Interconnections with n := 3 and n := 128 for AURIX

21.7.2 MultiCAN+ Module External Registers

The registers listed in **Figure 21-29** are not included in the MultiCAN+ module kernel, some registers must be programmed for proper operation of the MultiCAN+ module.

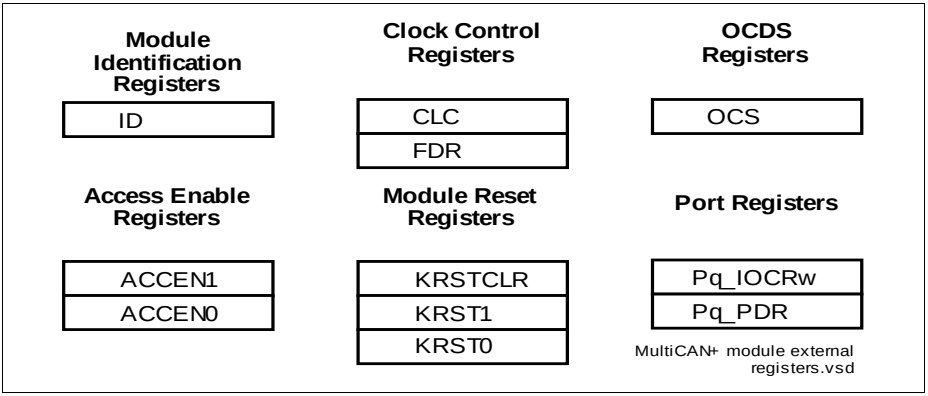


Figure 21-29 CAN Implementation-specific Special Function Registers

Table 21-20 MultiCAN+ **Module External Registers**

Short Name	Description	Offset Addr	Access Mode		Reset Class	Description see
			Read	Write		
Module Identification Registers						
ID	Module Identification Register	008_H	U, SV	nBE	Application Reset	**Page 21-79**
Clock Control Registers						
CLC	Clock Control Register	000_H	U, SV	SV, E, P	Application Reset	**Page 21-167**
FDR	Fractional Divider Register	$00C_H$	U, SV	SV, E, P	Application Reset	**Page 21-168**
OCDS Registers						
OCS	OCDS Control and Status Register	$0E8_H$	U, SV	SV, P	1	**Page 21-155**
Module Reset Registers						

Table 21-20 MultiCAN+ **Module External Registers** (cont'd)

Short Name	Description	Offset Addr	Access Mode		Reset Class	Description see
			Read	Write		
KRSTCLR	Reset Status Clear Register	$0EC_H$	U, SV	SV, E, P	3	Page 21-162
KRST1	Reset Control Register 1	$0F0_H$	U, SV	SV, E, P	3	Page 21-161
KRST0	Reset Control Register 0	$0F4_H$	U, SV	SV, E, P	3	Page 21-159
Access Enable Registers						
ACCEN1	Access Enable Register 1	$0F8_H$	U, SV	SV, SE	3	Page 21-158
ACCEN0	Access Enable Register 0	$0FC_H$	U, SV	SV, SE	3	Page 21-157

21.7.2.1 System Registers

OCDS Trigger Bus (OTGB)

The OCDS Control and Status (OCS) register is cleared by Debug Reset. The OCS register can only be written when the OCDS is enabled. If OCDS is being disabled, the OCS register value will not change. When OCDS is disabled the OCS suspend control is ineffective. Write access is 32bit wide only and requires Supervisor Mode.

OCS
OCDS Control and Status **($0E8_H$)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		SUS STA	SUS _P	SUS				0							
r		rh	w	rw				r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												TG _P	TGB	TGS	
r												w	rw	rw	

Field	Bits	Type	Description
TGS	[1:0]	rw	**Trigger Set for OTGB0/1** 0_H No Trigger Set output 1_H TS16_CAN **others**, reserved
TGB	2	rw	**OTGB0/1 Bus Select** 0_B Trigger Set is output on OTGB0 1_B Trigger Set is output on OTGB1
TG_P	3	w	**TGS, TGB Write Protection** TGS and TGB are only written when TG_P is 1, otherwise unchanged. Read as 0.

Field	Bits	Type	Description
SUS	[27:24]	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0_H Will not suspend 1_H Hard suspend. Clock is switched off immediately. Do not use this mode in normal CAN applications, this mode is meant for debugging the peripheral IP. 2_H Soft suspend of selected CAN nodes. Individually controlled with NCRx.SUSEN. **others**, reserved
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B CAN nodes are not (yet) suspended 1_B All selected (SUS) CAN nodes are suspended
0	[23:4], [31:30]	r	**Reserved** Read as 0; must be written with 0.

Access Enable Register 0

The Access Enable Register 0 controls write access[1)] for transactions with the on chip bus master TAG ID 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000_B, EN1 -> TAG ID 000001_B,.. ,EN31 -> TAG ID 011111_B.

ACCEN0
Access Enable Register 0 ($0FC_H$) Reset Value: FFFF FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN 31	**EN 30**	**EN 29**	**EN 28**	**EN 27**	**EN 26**	**EN 25**	**EN 24**	**EN 23**	**EN 22**	**EN 21**	**EN 20**	**EN 19**	**EN 18**	**EN 17**	**EN 16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN 15	**EN 14**	**EN 13**	**EN 12**	**EN 11**	**EN 10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n = 0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

1) The BPI_FPI Access Enable functionality controls only write transactions to the CLC, OCS, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers

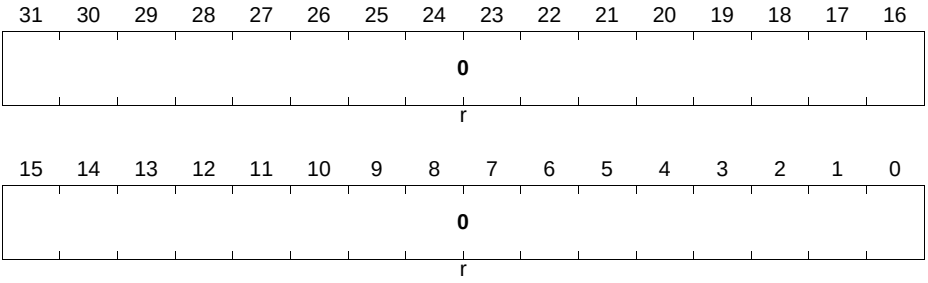
Access Enable Register 1

The Access Enable Register 1 controls write access[1)] for transactions with the on chip bus master TAG ID 100000_B to 111111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. ACCEN0 is not implemented with register bits as the related TAG IDs are not used in the AURIX devices.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001B, ... ,EN31 -> TAG ID 111111B.

ACCEN1

Access Enable Register 1 **($0F8_H$)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
0	[31:0]	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 0 (KRST0)

The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers related to the module kernel that should be reset (kernel 0 or kernel 1).The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing to it with ´1´.

Note: During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

KRST0
Kernel Reset Register 0 **($0F4_H$)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														RSTSTAT	RST
r														rh	rwh

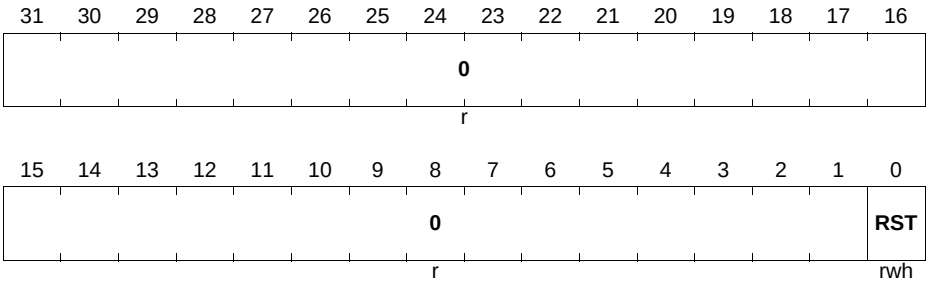
Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed.

Field	Bits	Type	Description
RSTSTAT	1	rw	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. 0_B No kernel reset was executed 1_B Kernel reset was executed This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register.
0	[31:2]	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1

The Kernel Reset Register 1 is used to reset the related module kernel. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers (CAN_KRSTx1.RST and CAN_KRSTx0.RST) related to the module kernel that should be reset (kernel 0 or kernel 1). The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

KRST1
Kernel Reset Register 1 **($0F0_H$)** **Reset Value: $0000\ 0000_H$**

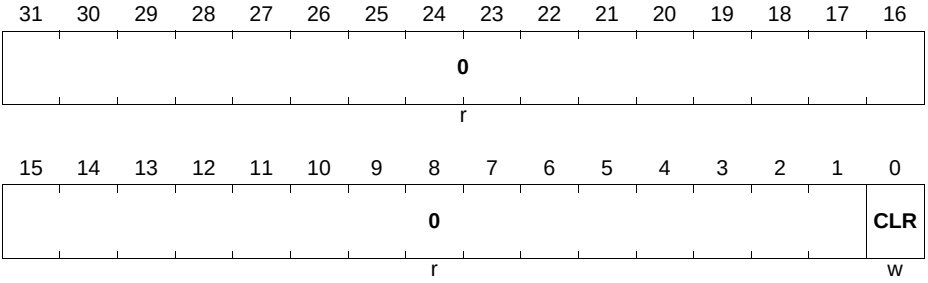


Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Clear Register

The Kernel Reset Clear Register is used to clear the Kernel Reset Status bit (CAN_KRST0.RSTSTAT).

KRSTCLR
Kernel Reset Status Clear Register (0EC$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT Read always as 0.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

21.7.3 MultiCAN+ Clock Interconnects With SCU

The MultiCAN+ module clock inputs are connected to System Control Unit (SCU) clock control unit (CCU) as shown in **Figure 21-30** and **Table 21-21** below.

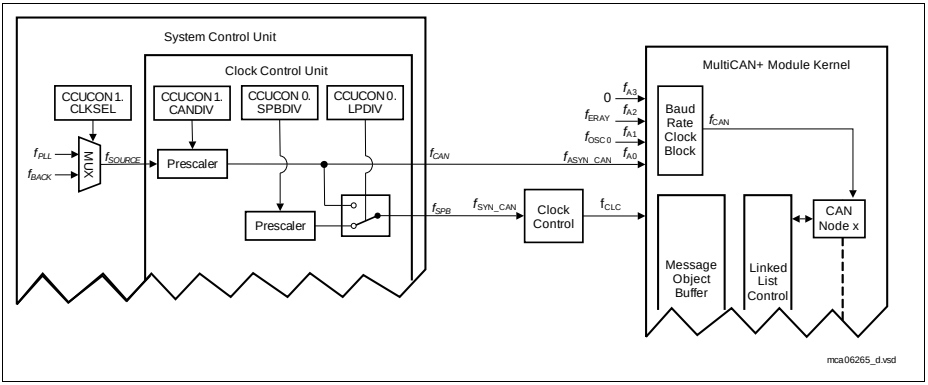


Figure 21-30 MultiCAN+ Clock Interconnects with SCU

Table 21-21 MultiCAN+ Clock Interconnects

CAN Clock Inputs	Connected to	Description
f_{ASYN_CAN}	SCU	f_{ASYN_CAN} of the MultiCAN+ module is one of the clock inputs of the Baud Rate Clock Block, providing the MultiCAN+ module timer clock f_{CAN}. f_{ASYN_CAN} connects to SCU CCU f_{CAN} and is controlled by CCUCON1.CANDIV. The CCUCON1.CANDIV is able to select different values of prescalers based on a selectable f_{PLL} or f_{BACK} source.
f_{SYN_CAN}	SCU	f_{SYN_CAN} of the MultiCAN+ module is the clock input of the Clock Control Register Block, providing the main kernel clock f_{CLC}. f_{SYN_CAN} connects to SCU CCU and is multiplexed between f_{SPB} and f_{CAN}. In normal mode, f_{SYN_CAN} is connected to f_{SPB} where else in pretended networking mode (CCUCON0.LPDIV >0), f_{SYN_CAN} connects to f_{CAN} of SCU CCU.

Note: The f_{CAN} mentioned on the SCU chapter does not refer to the same f_{CAN} that is mentioned on the MultiCAN+ chapter.

9. *The f_{CAN} mentioned on the SCU chapter refers in general to connection for MultiCAN+ clocking inputs f_{ASYN_CAN} and f_{SYN_CAN}. Depending on the settings of CCUCON1.CANDIV and CCUCON0.LPDIV different clock sources are connected (See **Table 21-21**).*
10. *The f_{CAN} mentioned on the MultiCAN+ chapter refers to the module timer clock (See **Section 21.7.4.2**).*

21.7.4 Module Clock Generation

This chapter describes the way the module get's its clock.

21.7.4.1 Clock Selection

The bit timing machine and the rest of the MultiCAN+ module are separate frequency domains and can be driven by separate independent frequencies. The bit timing unit can be driven by the SPB bus clock or with the direct oscillator clock, and the rest of the chip is driven only by the SPB bus clock.

The purpose of supplying the bit timing unit with a direct oscillator clock is to avoid the clock jitter added by the PLL, necessary when the chip is driven by a low cost ceramic resonator instead of by high precision quartz crystal.

Selecting the clock source for the bit timing unit is done by programming the bit-field MCR.CLKSEL.

Enabling and disabling the clock of the module by using CLC.DISR affects always both frequency domains, so that when f_{CLC} is switched off, f_A is also switched off.

21.7.4.2 Fractional Divider

As shown in **Figure 21-31**, the clock signals for the MultiCAN+ module are generated and controlled by a clock control unit. This clock generation unit is responsible for the enable/disable control, the clock frequency adjustment, and the debug clock control. This unit includes two registers:

- CAN_CLC: generation of the module control clock f_{CLC}
- CAN_FDR: frequency control of the module timer clock f_{CAN}

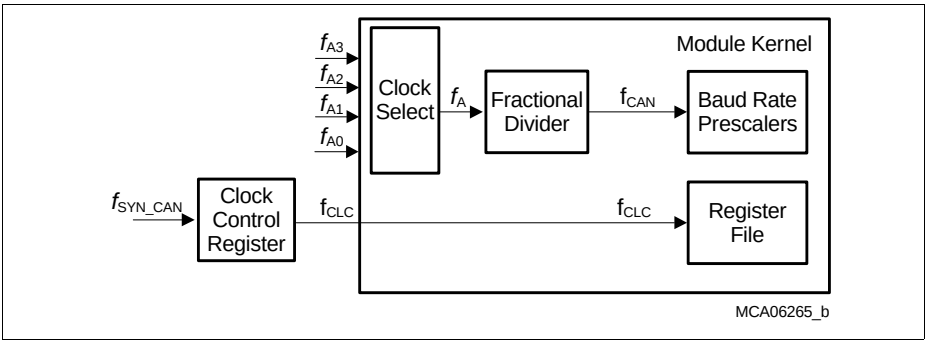


Figure 21-31 MultiCAN+ Module Clock Generation

The f_{SYN_CAN} is identical to f_{SPB}.

The module control clock f_{CLC} is used inside the MultiCAN+ module for control purposes such as clocking of control logic and register operations. The frequency of f_{CLC} is identical to the system clock frequency f_{SPB}. The clock control register CAN_CLC makes it possible to enable/disable f_{CLC} under certain conditions.

The module timer clock f_{CAN} is used inside the MultiCAN+ module as input clock for all timing relevant operations (e.g. bit timing). The settings in the CAN_FDR register determine the frequency of the module timer clock f_{CAN} according the following two formulas:

$$f_{CAN} = f_A \times \frac{1}{n} \quad \text{with n = 1024 - CAN_FDR.STEP} \tag{21.2}$$

$$f_{CAN} = f_A \times \frac{n}{1024} \quad \text{with n = 0-1023} \tag{21.3}$$

Equation (21.2) applies to normal divider mode (CAN_FDR.DM = 01_B) of the fractional divider. **Equation (21.3)** applies to fractional divider mode (CAN_FDR.DM = 10_B).

Note: The CAN module is disabled after reset. In general, after reset, the module control clock f_{CLC} must be switched on (writing to register CAN_CLC) before the frequency of the module timer clock f_{CAN} is defined (writing to register CAN_FDR).

CAN Clock Control Register

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI for the module. Where a module kernel is connected to the CLC clock control interface, CLC controls the f_{CAN} module clock signal, sleep mode and fast shut-off mode for the module.

CLC
CAN Clock Control Register (000_H) Reset Value: 0000 0003$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												EDIS	0	DISS	DISR
r												rw	r	rh	rw

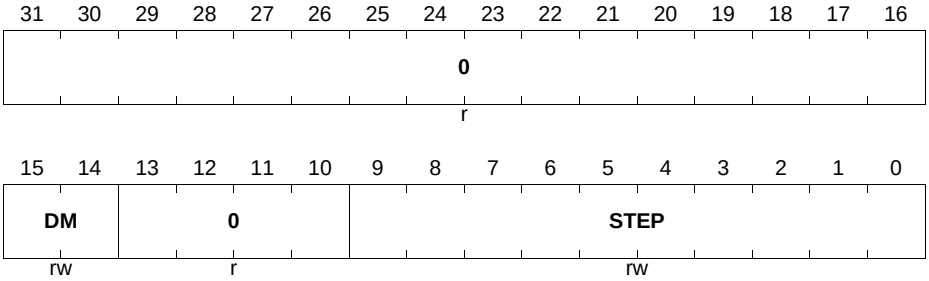
Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. Note that no register access is possible to any register while module is disabled.
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module.
EDIS	3	rw	**Sleep Mode Enable Control** Used to control module's sleep mode.
0	[31:4], 2	r	**Reserved** Read as 0; should be written with 0.

Note: The number of module clock cycles (wait states) which are required by the kernel to execute a read or write access depends on the selected CLC clock frequency.

The fractional divider register allows the programmer to control the clock rate of the module timer clock f_{CAN}.

FDR

CAN Fractional Divider Register ($00C_H$) Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
STEP	[9:0]	rw	**Step Value** Reload or addition value for the result.
DM	[15:14]	rw	**Divider Mode** This bit field selects normal divider mode, fractional divider mode, and off-state.
0	[13:10], [31:16]	r	**Reserved** Read as 0; should be written with 0.

21.7.5 Port and I/O Line Control

The interconnections between the MultiCAN+ module and the port I/O lines are controlled in the port logic. Additionally to the port input selection, the following port control operations must be executed:

- Input/output function selection (IOCR registers)
- Pad driver characteristics selection for the outputs (PDR registers)

21.7.5.1 Input/Output Function Selection in Ports

The port input/output control registers contain the bit fields that select the digital output and input driver characteristics such as pull-up/down devices, port direction (input/output), open-drain, and alternate output selections. The I/O lines for the MultiCAN+ module are controlled by the port input/output control registers, which are described in the port chapter. In case of discrepancies between port chapter and CAN chapter, the description in the port chapter is correct.

Table 21-22 shows the corresponding pins, sorted by node. Even though the table is pair wise, it is possible to select a different pairing for RXD and TXD. In addition the RXSEL value to be programmed for the Receive Pins is part of this table. For more information on RXSEL please see **Chapter 21.7.5.2**.

Table 21-22 MultiCAN+ I/O Control Selection and Setup for TC23x

Node	RXD	NPCRx.RXSEL	TXD
Nodes of CAN_			
CAN0	P2.1 / RXDCAN0A	000_B	P2.0 / TXDCAN0[1)]
	P20.7 / RXDCAN0B	001_B	P20.8 / TXDCAN0
	P2.4 / RXDCAN0D	011_B	P2.5 / TXDCAN0[1)]
	P33.7 / RXDCAN0E	100_B	P33.8 / TXDCAN0[1)]
	P34.2 / RXDCAN0G	110_B	P34.1 / TXDCAN0
	Node1		
CAN1	P15.3 / RXDCAN1A	000_B	P15.2 / TXDCAN1
	P14.1 / RXDCAN1B	001_B	P14.0 / TXDCAN1[1)]
	P0.1 / RXDCAN1D	011_B	P0.0 / TXDCAN1
CAN2	P15.1 / RXDCAN2A	000_B	P15.0 / TXDCAN2
	P2.3 / RXDCAN2B	001_B	P2.2 / TXDCAN2[1)]
	P14.8 / RXDCAN2D	011_B	
	P10.2 / RXDCAN2E	110_B	P10.3 / TXDCAN2
Nodes of CAN1_			

Table 21-22 MultiCAN+ I/O Control Selection and Setup for TC23x (cont'd)

Node	RXD	NPCRx.RXSEL	TXD
CAN0	P10.5 / RXDCAN0A	000_B	P10.6 / TXDCAN0
	P13.1 / RXDCAN0B	001_B	P13.0 / TXDCAN0
	P2.6 / RXDCAN0F	101_B	P2.4 / TXDCAN0[1)]
	P0.5 / RXDCAN0G	110_B	P0.4 / TXDCAN0
	P33.6 / RXDCAN0H	111_B	P33.7 / TXDCAN0[1)]
CAN1	P0.3 / RXDCAN1A	000_B	P0.7 / TXDCAN1
	P13.3 / RXDCAN1B	001_B	P13.2 / TXDCAN1
	P20.0 / RXDCAN1C	010_B	
	P2.7 / RXDCAN1F	101_B	P2.5 / TXDCAN1[1)]
	P0.6 / RXDCAN1G	110_B	
	P20.12 / RXDCAN1H	111_B	P20.11 / TXDCAN1[1)]
CAN2	P0.3 / RXDCAN2A	000_B	P0.2 / TXDCAN2
	P20.0 / RXDCAN2C	010_B	P20.3 / TXDCAN2
	P11.10 / RXDCAN2D	011_B	
	P20.9 / RXDCAN2E	100_B	P20.10 / TXDCAN2
	P0.8 / RXDCAN2G	110_B	P0.9 / TXDCAN2
	P20.13 / RXDCAN2H	111_B	P20.14 / TXDCAN2[1)]

1) These ports are using fast pads and are suitable for TxD outputs with CAN FD at higher baud rates.

21.7.5.2 Node Receive Input Selection

Additionally to the I/O control selection, as defined in the port chapter, the selection of a CAN node's receive input line requires that bit field RXSEL in its node port control register NPCRx must be set according to **Table 21-22**. Values for NPCRx.RXSEL other than those of the table mentioned above will result in a recessive receive input for node x. As a hint A results in 0x0, B in 0x1 until H resulting in the value of 0x7.

This feature allows, for example, a CAN node which operates in analyzer mode to monitor the receive operations of its neighbor CAN node.

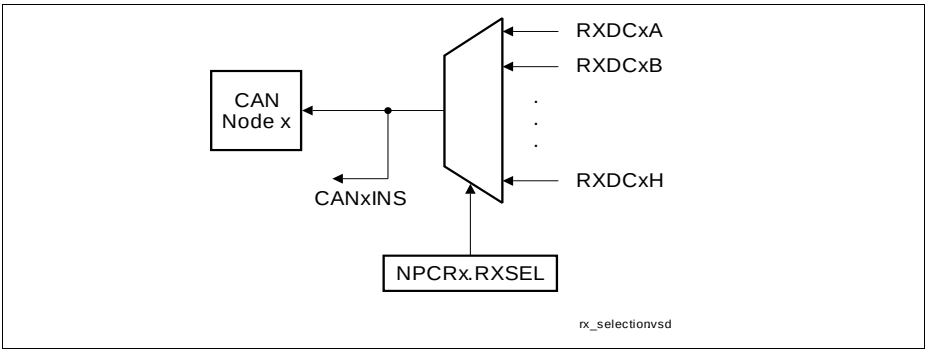


Figure 21-32 CAN Module Receive Input Selection

21.7.5.3 CAN Transmit Trigger Inputs

The CAN transmit trigger inputs of MultiCAN and MultiCAN1 are used to trigger for transmission of a message. In the TC21x/TC22x/TC23x, these input lines are connected as shown in **Table 21-23**.

Table 21-23 CAN Transmit Trigger Inputs

Receive Input	Connected to	From / to Module
CAN Node 0		
STM_N0_TRIG	stm0.systim_irq_o(0)	System Timer Module (STM)
GTM_N0_TRIG	gtm.can_trig_0_o	General Timer Module (GTM)
...		
...	...	...
CAN Node 2		
STM_N2_TRIG	stm0.systim_irq_o(0)	System Timer Module (STM)
GTM_N2_TRIG	gtm.can_trig_2_o	General Timer Module (GTM)
CAN1 Node 0		
STM_N0_TRIG	stm0.systim_irq_o(0)	System Timer Module (STM)
GTM_N0_TRIG	gtm.can_trig_0_o	General Timer Module (GTM)
CAN1 Node 1		
STM_N1_TRIG	stm0.systim_irq_o(0)	System Timer Module (STM)
GTM_N1_TRIG	gtm.can_trig_1_o	General Timer Module (GTM)
CAN1 Node 2		

Table 21-23 CAN Transmit Trigger Inputs (cont'd)

Receive Input	Connected to	From / to Module
STM_N2_TRIG	stm0.systim_irq_o(0)	System Timer Module (STM)
GTM_N2_TRIG	gtm.can_trig_2_o	General Timer Module (GTM)

21.7.5.4 Connections to Interrupt Router Inputs

The interrupt output line INT_O0-15 of MultiCAN and INT_O0-7 of MultiCAN1 is connected to the Interrupt Router module, see **Table 21-24** and **Table 21-25**.

Table 21-24 Interrupt Router Inputs

Interrupt Router Input	Connected to CAN Interrupt Output
SRC_CANINT0	INT_O0
SRC_CANINT1	INT_O1
SRC_CANINT2	INT_O2
SRC_CANINT3	INT_O3
SRC_CANINT4	INT_O4
SRC_CANINT5	INT_O5
SRC_CANINT6	INT_O6
SRC_CANINT7	INT_O7
SRC_CANINT8	INT_O8
SRC_CANINT9	INT_O9
SRC_CANINT10	INT_O10
SRC_CANINT11	INT_O11
SRC_CANINT12	INT_O12
SRC_CANINT13	INT_O13
SRC_CANINT14	INT_O14
SRC_CANINT15	INT_O15

Table 21-25 Interrupt Router Inputs (CAN1**)**

Interrupt Router Input	Connected to CAN1 Interrupt Output
SRC_CAN1INT0	INT_O0
SRC_CAN1INT1	INT_O1
SRC_CAN1INT2	INT_O2

Table 21-25 Interrupt Router Inputs (CAN1) (cont'd)

Interrupt Router Input	Connected to CAN1 Interrupt Output
SRC_CAN1INT3	INT_O3
SRC_CAN1INT4	INT_O4
SRC_CAN1INT5	INT_O5
SRC_CAN1INT6	INT_O6
SRC_CAN1INT7	INT_O7

21.7.5.5 Connections to General Timer Module (GTM) Inputs

The interrupt output line INT_O12-15 of MultiCAN+ is connected to the General Timer Module, see **Table 21-26**.

Table 21-26 General Timer Module Inputs

GTM Input	Connected to CAN Interrupt Output
Timer input (gtm.tim_0_muxin_1_i(13))	INT_O12 (mcan0.int_req_o(12))
Timer input (gtm.tim_0_muxin_2_i(13))	INT_O13 (mcan0.int_req_o(13))
Timer input (gtm.tim_0_muxin_3_i(13))	INT_O14 (mcan0.int_req_o(14))
Timer input (gtm.tim_0_muxin_4_i(13))	INT_O15 (mcan0.int_req_o(15))

21.7.6 Interrupt Control

The interrupt control logic in the MultiCAN+ module uses an interrupt compressing scheme that allows high flexibility in interrupt processing. There are hardware and software interrupt sources available:

- CAN node interrupts:
  - Five different interrupt sources for each of the 3 CAN nodes = 5 * 3 interrupt sources (CAN) or 3 CAN nodes = 5 * 3 interrupt sources (CAN1)
- Message object interrupts:
  - Two interrupt source for each message object = 2 * 128 interrupt sources (CAN) or 2 * 128 interrupt sources (CAN1)
- One register (MITR) to initiate 16 (CAN) or 8 (CAN1) interrupts via software

Each of the hardware initiated interrupt sources is controlled by a 4-bit interrupt pointer that directs the interrupt source to one of the 16 (CAN) or 8 (CAN1) interrupt outputs INT_Om (m = 0-15), INT_Om(m=0-8). This makes it possible to connect more than one interrupt source (between one and all) to one interrupt output line. The interrupt wiring matrix shown in **Figure 21-33** is built up according to the following rules:

- Each output of the 4-bit interrupt pointer demultiplexer is connected to exactly one OR-gate input of the INT_Om line. The number “m” of the corresponding selected INT_Om interrupt output line is defined by the interrupt pointer value.
- Each INT_Om output line has an input OR gate which is connected to all interrupt pointer demultiplexer outputs which are selected by an identical 4-bit pointer value.

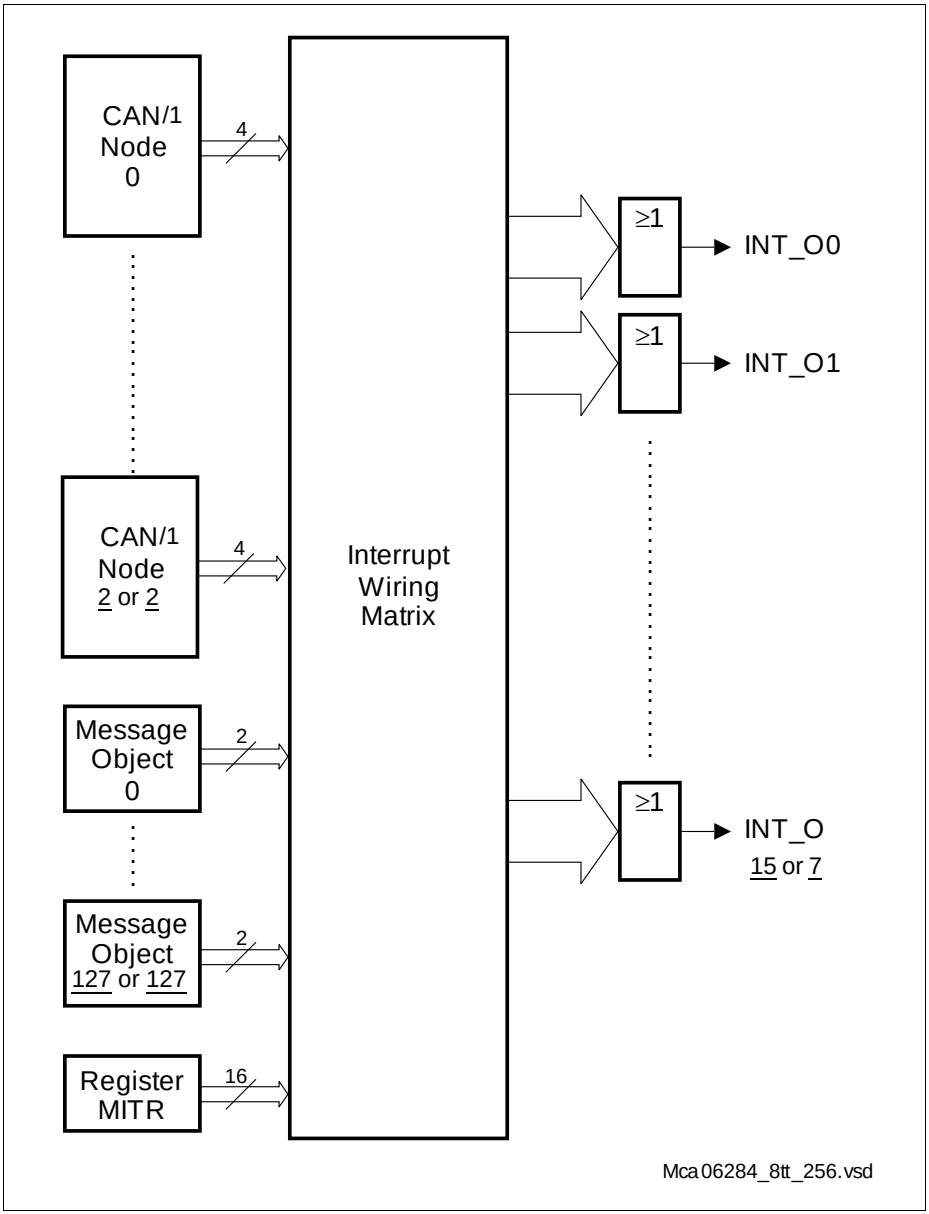


Figure 21-33 Interrupt Compressor

21.7.7 MultiCAN+ Module Register Address Map

The complete MultiCAN+ module register address map of **Figure 21-34** also shows the general implementation-specific registers for clock control, module identification, interrupt service request control and the absolute address information.

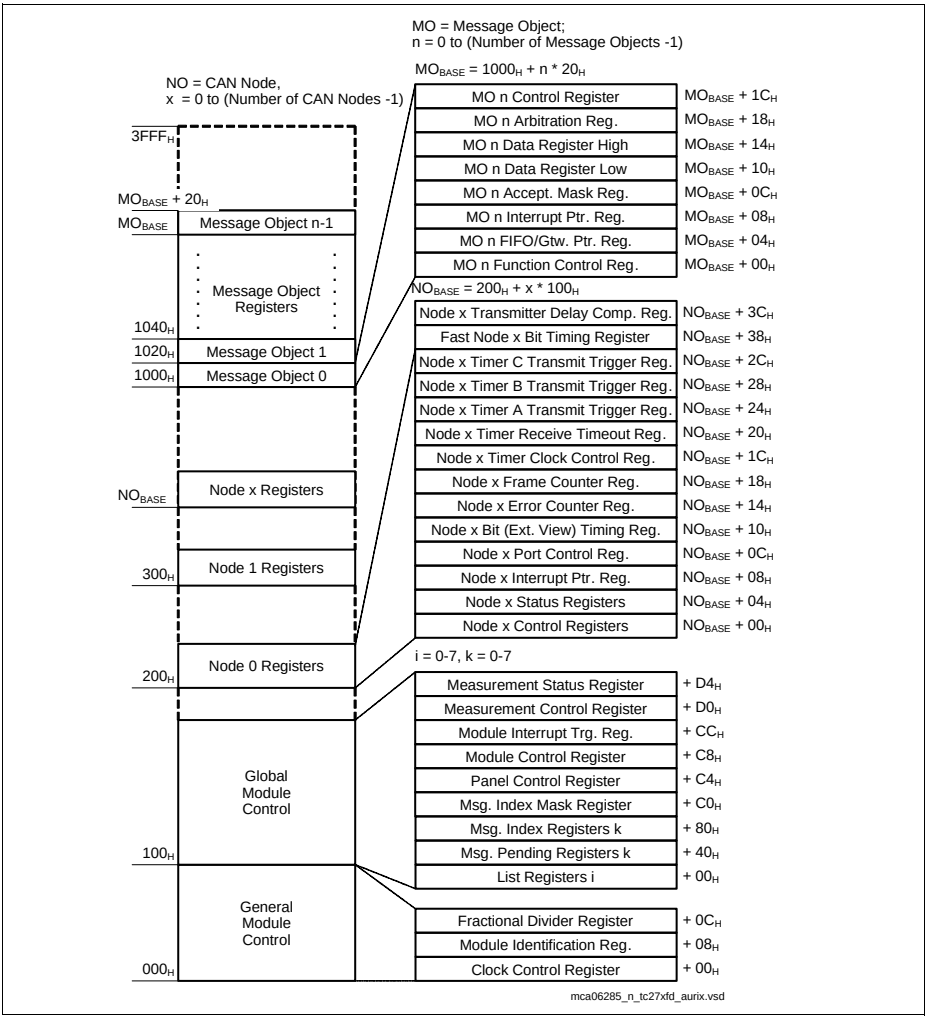


Figure 21-34 MultiCAN+ Register Address Map including CAN FD

21.8 Revision history

External revision history only shows major changes of MultiCAN+ module versions.

External Revision History (from MultiCAN+ v3.0 to MultiCAN+ v3.1)

- As per review meeting on 4Sep12 with designer, Removed for below:
  - Section 1.1.3.5 on Intermission removed as same as CAN V2.0B Bosch specs, under Section 9.1 Protocol Modifications, point 2.
  - Section 1.1.3.5 on Suspend transmission removed as same as CAN V2.0B Bosch specs, under Section 3.1.5 Interframe Spacing, Suspend Transmission.
  - Section 1.1.4.1 on Resynchronization removed as same as CAN V2.0B Bosch specs, under Section 8 Bit timing requirements, Resynchronization.
  - Section 1.1.4.1 on Synchronization rules, removed point 1,2 and 4. Point 3 and 5 are kept as new requirements in CAN FD.
  - Section 1.5.2 on Integrating mode, removed as same as CAN V2.0B Bosch specs, under Section 2 Sleep Mode / Wake-up.
- As per review meeting on 4Sep12 with designer, Added for below:
  - Additional text under MOFCRn.DLC, “Also if DLC>8 is programmed for message transmission, message transmit will be disabled.”
- As per review meeting on 4Sep12 with designer, Modified on Section 1.6.14 Debug over CAN with additional information.
- Added at Section 1.6.4 OCDS suspend, Note on “During suspend mode, kernel registers may be accessed only when the kernel clock is running”

External Revision History (from MultiCAN+ v3.1 to MultiCAN+ v3.2)

- As per AURIX JF on 1oct12, after bosch meeting on discussion of baudrates, request for below:
  - Modified on Section 1.1.3.1 on Fig1-2 CAN FD data frames changed to 64 data bytes and removed footnote on “Current design supports up to 8 data bytes payload only”
  - Removed on Section 1.3.1 footnote on “supports message transfer of data length code 0 to 8 data bytes with no Transmitter Delay compensation”
  - Removed on Section 1.5 “8 data bytes, no Transmitter Delay compensation “
  - Modified on Section 1.5.1, table 1-3, point 4 on Transmitter Delay compensation from “not implemented” to “implemented”
  - Removed on Section 1.6.8.1 on CAN Restricted operation mode.
  - Modified on MOFCR.DLC bit to support for more than 8 data bytes, have included 12, 16, 20, 24, 32, 48, 64 data bytes option.
  - Added Section 1.5.2 Transmitter Delay Compensation and FNBTRx.TDC, FNBTRx.TDCO, FNBSTATx.TDCV.
- As per EMail on DXCM DXCM_DAP_over_CAN_messages_120822.pdf, added for below:

- Modified MCR.DXCEN to MCR.DXCM, DAP Over CAN Message Enable
- Added to bit description “If enabled the highest two message objects are reserved for debugger communication. DXCM is described in detail in the OCDS chapter”
- Modified conditional text settings for DAP over CAN to “Has_DXCM”.

External Revision History (from MultiCAN+ v3.2 to MultiCAN+ v3.3)

- Modified MultiCAN+ clocking from SCU for below
  - Various diagrams and text that refers “$f_{MULTICAN}$” to “f_{ASYN_CAN}” and “f_{SYN_CAN}”. Also added Section 1.13.3 MultiCAN+ Clock Interconnects with SCU
  - Section 1.6.2 Table 1-5 footnote changed from “To guarantee the minimum operating frequencies when low power mode is enabled, fCLC is switched in fMULTICAN to a operating clock source, when SCU register CCUCON.” to “To guarantee the minimum operating frequencies when low power mode (pretended networking mode) is enabled (i.e when frequency of fSPB is reduced), fSYN_CAN and fCLC are switched from fSPB to fCAN when SCU register CCUCON0.LPDIV>0.”
- Renamed Fast Node x Bit Timing Status Register to Node x Transmitter Delay Compensation Register and moved FNBTRx.TDC, FNBTRx.TDCO to NTDCRx.TDC, NTDCRx.TDCO.

External Revision History (from MultiCAN+ v3.3 to MultiCAN+ v3.4)

- As per mail on “Widening of NBTR register field while maintaining compatibility and more” modified for below:
-
  - Rearranged FNBTRx register bits to use same layout as NBTRx for lowest 16bits and empty for upper 32bits
  - Changed address of FNBTRx to 0x0238 and NTDCRx to 0x23C
  - Swapped on NTDCRx register bits on TDC =Bit 15 and TDCO= Bits 11-8
  - Added NBTEVRx, Node Bit Timing Extended View Register. NBTEVRx view applies when NCRx.FDEN=1 and NBTRx view applies when NCRx.FDEN=0.
  - NBTEVRx. TSEG1 is 6 bits, valid values from 2 to 63
  - NBTEVRx. TSEG2 is 4 bits, valid values are 1 to 15
  - NBTEVRx.SJW is 4 bits, valid values are 1 to 15

External Revision History (from MultiCAN+ v3.4 to MultiCAN+ v3.5)

- As per review feedback on V3.3 CANFD specs review modified for below:
  - Removed on Section1.1.3.1 text on point 2e “In current CAN FD implementation, up to 8 data bytes payload are supported only”
  - Added on Table1-4 Minimum Operating Frequencies, footnote “To guarantee the minimum operating frequencies when low power mode.”
  - Removed on Section 1.4.12.4 “vice versa”

  - Update of Table 1-24 MultiCAN+ Clock Interconnects
- Additional modifications for below:
  - Added on NCRx.FDEN footnote on “Message transmission in Long & Fast CAN FD Frames. Message Reception according to CAN FD Protocol Specification, (i.e Able to Receive Classical CAN Format Frames ISO11898-1 and Long + Fast CAN FD Frames)”
  - Added on Table 1-8 Coding of DLC in CAN FD and MOFCRx.DLC
  - Added MOFCR.FDF, MOFCR.BRS bit to indicate message format reception when MOFCRx.MOFM =1 (CAN FD enabled) (i.e received frame in classical CAN or CAN FD format)
  - Added on NSRx.FLEC, fast last error code to indicate the error code during the fast bit rate.
  - Added note on Table 1-13 Encoding of LEC bit field, “When a frame in CAN FD format reached data phase with BRS flag set, the next CAN event (error/valid frame) will be shown in FLEC instead of LEC. An error in a fixed stuff bit of CAN FD CRC will be shown as Form and not Stuff Error.”
- As per review feedback on V3.3 CANFD specs review modified for below:
  - Added description of message object used for DXCM RX and TX.
  - Used DXCM ITS text for DXCM message data description.

External Revision History (from MultiCAN+ v3.5 to MultiCAN+ v3.6)

- As per mail “C3 preparation for MultiCAN+” modified for Section 1.6.14 Debug Over CAN, to a summarized short version.
- As per mail “MultiCAN ITS problem DXCM bit position” modified for MCR.DXCM to bit 8.

External Revision History (from MultiCAN+ v3.6 to MultiCAN+ v3.7)

- As per mail on “Message Mode proposal for CAN FD 64 data bytes” modified for below.
  - Added **Section 21.4.12.10** on CAN FD 64 byte Message Mode
  - Added **CAN_EMOzDATA0 (z = 0-127)** Extended Message Object Data Register and on Table 1-11
  - Modified on MOFCR.MMC bit 101 to CAN FD 64 bytes message mode.
- As per TC24x_tag_coverage.xls added tags AURIX_GEN1:5246, 5396, 5373, 5374:
- As per AI124647 added additional CAN1 (3nodes/128MO) on module base address, modified Fig1-49 CAN Module Implementation and Interconnections and various places for additional CAN1 node.
- As per CAN FD weekly meeting on 21Nov12, modified for MOFCR.BRS and MOFCR.FDF bit from rh to rwh.

External Revision History (from MultiCAN+ v3.7 to MultiCAN+ v3.8)

- Added note on **Section 21.4.6.5** “The transmit request bits (i.e NTATTx.TXMO / NTBTTx.TXMO / NTCTTx.TXMO) in the timer control register of a node is able to trigger transmit request (MOSTATn.TXRQ) of message object in a list belonging to any other CAN nodes”.
- As per MultiCAN+ V3.7 feedback review
  – Modified on CAN_NBTEVRx text to “NBTEVRx contains all parameters to setup CAN bit timing for Nominal Bit Rate”.
  – Modified on CAN_FNBTRx text to “FNBTRx contains all parameters to setup CAN bit timing for Data Bit Rate”.
  – Modified on CAN_FNBTRx register bits name to FTSEG1, FTSEG2, FSJW and FBRP.
  – Modified on CAN_NBTEVRx.TSEG2 to 5bits, bit20 to bit16.

External Revision History (from MultiCAN+ v3.8 to MultiCAN+ v3.9)

- As per MultiCAN+ V3.7 feedback review.
  – As per mail “CCE protection of register tdcr” added on CAN_NTDCR register “NTDCRx register can be written only if bit NCRx.CCE is set.”. Removed on NTDCR.TDC text on “this bit is CCE and INIT protected”.
  – Added on soft configuration section a footnote on CAN FD “When CAN FD is disabled by PRDCFG, access to CAN FD registers do not generate a bus error. To verify CAN FD is disabled, try to set NCRx.FDEN bit and read back the NCRx register values, the NCRx.FDEN bits remain cleared.”
  – Modified on NTDCR.TDCV text from fcan/fclc to time quanta “Valid values for TDCV are 0 to 31 times of time quanta t_Q”
  – Modified on NTDCR.TDCO text from facn/fclc to time quanta “Valid values for TDCO are 0 to 15 times of time quanta t_Q.”
  – Added text under Section CAN FD 64 bytes Message Mode “When MSGVAL of the base object is cleared, an ongoing data transfer to TOP and BOT objects are aborted, thereby freeing objects BOT and TOP. Clearing MSGVAL bits of BOT, TOP objects and the extra message objects do not stop an ongoing data transfer and are not considered at all.”
  – Added text on NCRx.FDEN, MOFCR.MOFM, MOFCR.BRS and MOFCR.FDF bits “Please see Table 1-19 CAN FD Transmit and Receive Behavior.

External Revision History (from MultiCAN+ v3.9 to MultiCAN+ v3.10)

- As per MultiCAN+ V3.7 feedback review
  – Updated Table 1-19 CAN FD Transmit and Receive Behavior, corrected for transmit behavior “Classical CAN Frames” FDEN from 0_B to X_B.

External Revision History (from MultiCAN+ v3.10 to MultiCAN+ v3.11)

- As per MultiCAN+ V3.7 feedback review
  - Updated Table 1-30 MultiCAN+ I/O control Selection and Setup and Table 1-36 Receive Input Selection
  - Added Footnote on Table 1-30 “These ports are using A1+ pads and are suitable for TxD outputs with CAN FD at higher baud rates.”
  - Added Table 1-9 “Minimum Operating Frequencies for CAN FD [MHz]”
- This bulleted point is not applicable to TC23x.
- Updated on Section 1.6.2 Clock Control on example of equation 1.1 to “As an example, when f_{CLC} = 10MHz, f_{CAN} = 20MHz, No of active CAN nodes =2, Baudratemax=[(8x50ns) + (8x100ns) + (4x2x100ns)] = 2000ns = 500 KBaud
- Updated for TC23x Section 1.13.5.1 Input / Output Function Selection in Ports, Section 1.13.5.2 Node Receive Input Selection, Section 1.13.5.4 CAN Transmit Trigger Inputs, Fig 1-49 CAN Module Implementation and Interconnections.

External Revision History (from MultiCAN+ v3.11 to MultiCAN+ v3.12)

- Updated for TC23x on MOFCR register corrected BRS to bit5, FDF to bit6 and MOFM to bit 7.
- Modified on Section 1.1.3.2 Remote Frames, paragraph from “Remote frames is only defined in CAN format, Remote frames and RTR bit do not exist for CAN FD format. ~~In a CAN FD node, Remote frames request are always replied back in CAN format and will never receive replies in CAN FD format.~~” to “Remote frames is only defined in CAN format, Remote frames and RTR bit do not exist for CAN FD format. Remote frame requests do not change the format of the preconfigured reply data frames (i.e FDF and BRS bits of preconfigured data frames are not changed on remote frame requests, reply data frames may be in CAN base/extended or CAN FD base/extended).”
- Modified on Table 1-20 CAN FD Transmit and Receive Behavior on receive behavior to FDEN=0, MOFM=X, BRS=0/1, FDF=0/1- Classical CAN Frames, FDEN=1, MOFM=X, BRS=0/1, FDF=0/1- Classical CAN Frames, Long Frames, Long + Fast Frames.
- Added on Table 1-20 CAN FD Transmit and Receive Behavior, footnote on receive behavior “Please note that Remote Request Frames do not change BRS and FDF bits.”

External Revision History (from MultiCAN+ v3.12 to MultiCAN+ v3.13)

- Modified for below
  - Removed from Table 1-31 MultiCAN+ I/O Control Selection and Setup, incorrect RXDCAN1E-P20.9.
  - Added on MCR.CLKSEL bit option 0100_B E-Ray clock “(Feray)”.

  - Added Fig1-52 MultiCAN+ Interconnects with SCU on section 1.13.3 MultiCAN+ Clock Interconnects With SCU.
  - Corrected on Fig 19-27 CAN Module Implementation and Interconnections, change to 128 MO, Feray, corrected STM_Nx_TRIG and GTM_Nx_TRIG to CAN and CAN1 module.
- This bulleted point is not applicable to TC23x.

External Revision History (from MultiCAN+ v3.13 to MultiCAN+ v3.14)

- Removed NSRx.RESI bit would generate interrupt on NSRx.ALERT and NCRx.ALIE. Also modified NSRx.RESI bit to 'rh' changed description to "This bit is an error flag that is set when the ESI flag in a received CAN FD frame is set." and "0_b-Last received CAN FD message did not have its ESI flag set. 1_b-Last received CAN FD message had its ESI flag set."
- Addition to note on Table 1-13 Encoding of LEC bit field, "Correspondingly CAN event (error/valid frame) will be shown back at LEC after the first bit of the CRC delimiter when CAN FD switches from Data bit rate to Nominal bit rate".

External Revision History (from MultiCAN+ v3.14 to MultiCAN+ v3.15)

- Removal of MOFCR.MOFM bit. Update of Table 1-20 CAN FD Transmit and Receive Behaviour.
- Update of Table 1-5 Minimum Operating Frequencies for CAN FD [MHz].
  - Addition of Acceleration Factor rows for values from 1 to 8
  - Modified on Footnote 3) with addition of Base Nominal Bit Rate is 1Mbit/s and addition "when several nodes operate at different acceleration factor A.F, only the node operating the highest acceleration factor need to be considered for the required minimum operating frequency requirement."

External Revision History (from MultiCAN+ v3.15 to MultiCAN+ v3.16)

- Section 1.6.12.4 Message Object Format updated with removal of MOFCR.MOFM and replaced with description of NCRx.FDEN and MOFCRn.FDF, MOFCRn.BRS bits.
- CAN FD feature added for TC27x C-step, TC26x B-step, TC29x B-step.
- Updated on features list, with bullet item "Supports an acceleration factor of 8 when operating in CAN FD mode per Bosch CAN FD V1.0 April17th 2012"
- Debug over CAN feature removed for TC27x C-step, TC26x B-step, TC29x B-step.

External Revision History (from MultiCAN+ v3.16 to MultiCAN+ v3.17)

- Added on section 1.4.14 Debug over CAN and under features list for item on Debug over CAN, footnote "Debug over CAN feature is available through CAN module only and not available on CAN1 module."

- Updated Table 1-5 Minimum Operating Frequencies for CAN FD [MHz], with corrected values from multican_performance_canfd_16apr13.xls
  – Modified Acceleration Factor rows for values from 1 to 5 and at various places that mentions AF to 1 to 5.

External Revision History (from MultiCAN+ v3.17 to MultiCAN+ v3.18)

- Added additional text on Section 1.5.2 Transmitter Delay Compensation to describe more details on secondary sample point implementation.
- Added on footnote of Table 1-5 Minimum Operating Frequencies for CAN FD “When message objects are configured as transmit or receive FIFO structures i.e MOFCRn.MMC = 0001/0010, only the base object of 1 is counted towards the minimum frequency requirement, all other slave objects on the FIFO do not count towards the minimum frequency requirement. See Section 1.6.12.5.
- Relabel tags CAN_NUMBER_NODES to CAN1_NUMBER_NODES on Table1-2.

External Revision History (from MultiCAN+ v3.18 to MultiCAN+ v3.19)

- Text enhancements below:
  – Under Section 1.1.3.1, abbreviation added for FDF, BRS, ESI bit
  – Under Section 1.1.3.3, amended from “shall switch back” to “will switch back”
  – Under Section 1.3, amended from “on every nodes” to “on all nodes”
  – Under Section 1.3.1, amended from “measured Transmitter Delay” to measured loop delay”
  – Under CAN_MOFCR on CAN FD Transmit and Receive Behavior, added text “Message transmission and reception behavior of CAN data frames is summarized at Table 1-20 below.”
  – Under CAN_MOFGPR.TOP/BOT added text MOFGPR.BOT points to data byte 8-35 and MOFGPR.TOP points to 36-63.

External Revision History (from MultiCAN+ v3.19 to MultiCAN+ v3.20)

- All reference to EDL is renamed to FDF, register bit MOFCR.EDL is renamed to MOFCR.FDF, functionality remains the same.

External Revision History (from MultiCAN+ v3.20 to MultiCAN+ v3.21)

- This bulleted point is not applicable to TC23x.

External Revision History (from MultiCAN+ v3.21 to MultiCAN+ v3.22)

- Changed CAN FD descriptions to customer relevant information.

External Revision History (from MultiCAN+ v3.22 to MultiCAN+ v3.23)

- Corrected spelling bugs

External Revision History (from MultiCAN+ v3.23 to MultiCAN+ v3.24)

- Changes for XMC device frequencies, not applying for AURIX devices

External Revision History (from MultiCAN+ v3.24 to MultiCAN+ v3.25)

- Changes for XMC port naming. Correcting XMC specification by removing AURIX relevant issues.

External Revision History (from MultiCAN+ v3.25 to MultiCAN+ v3.26)

- Corrected memory space for XCM1000.
- Fixed pad naming for all devices.
- Corrected Safety tagging for AURIX
- CLC and FDR are existing on all modules, which are existent in one microcontroller.

External Revision History (from MultiCAN+ v3.26 to MultiCAN+ v3.27)

- Corrected interrupt mapping for XCM1000.

External Revision History (from MultiCAN+ v3.27 to MultiCAN+ v3.28)

- CRC issue of CAN FD documented.

External Revision History (from MultiCAN+ v3.28 to MultiCAN+ v3.29)

- MCR register, formula for clock switching mathematically not correct. As no clock cycles were calculated. Exchanged to clock cycles..

22 Single Edge Nibble Transmission (SENT)

This chapter describes the SENT Interface of the TC21x/TC22x/TC23x. It contains the following sections:

- Functional description of the SENT kernel (see **"Overview" on Page 22-3**)
- SENT kernel register descriptions (see **"SENT Kernel Registers" on Page 22-23**)
- TC21x/TC22x/TC23x implementation-specific details and registers of the SENT module (port connections and control, interrupt control, address decoding, and clock control,
  see **"SENT Module Implementation" on Page 22-71**

22.1 SENT Kernel Description

Figure 22-1 shows a global view of the SENT interface.

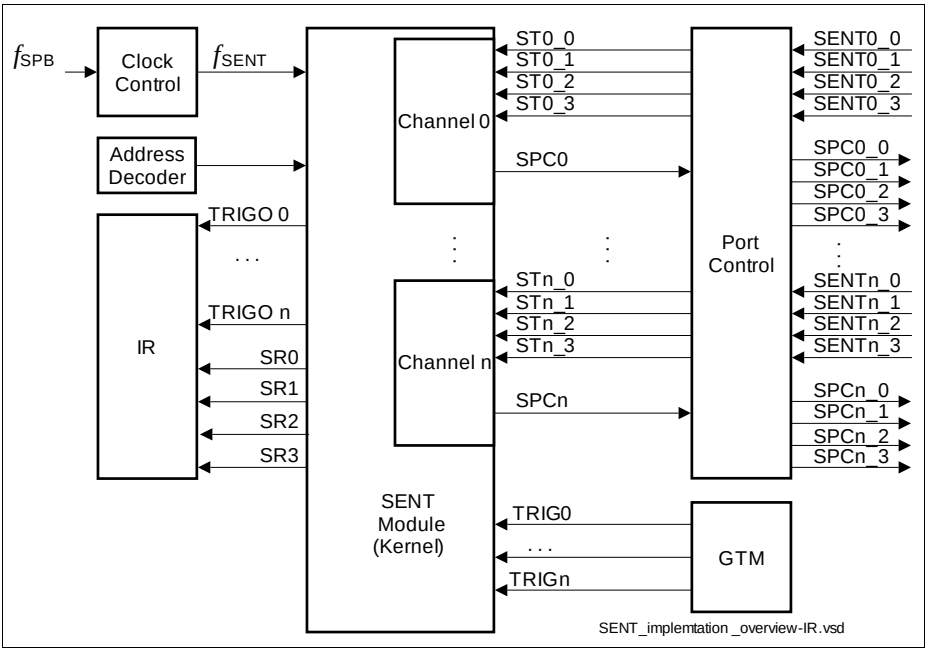


Figure 22-1 General Block Diagram of the SENT Interface

The SENT module communicates with the external world via one I/O line for each channel. The STx lines are the receive data input signals. They can overlay ADC inputs. If the optional SPC mode is used, they can be used on a port configured with an open drain transistor. This way the optional SPC data can be transmitted and the line is used

bidirectionally. In case of an external transceiver, receive and transmit path can be routed to two different ports.

22.1.1 Overview

The SENT interface provides a serial communication link typically used to connect sensors or other peripheral devices.

Clock control, address decoding, and service request control are managed by the SENT module kernel.

The SENT IP-module performs communication according to the SENT specification J2716 JAN2010.
While staying compliant to this standard, it is able to cover as well the Short PWM Code (SPC) protocol extensions. This enhances the standardized SENT protocol defined by J2716 JAN2010. SPC enables the use of enhanced protocol functionality like “synchronous”, “range selection” and “ID selection” protocol mode.

Receive data on a SENT channel can be set up according to the underlying application. In particular the number of nibbles forming one value is configurable.

The message storage consists of two 32-bit registers for each channel, representing a flexible double buffer system.

In SPC mode, maintaining the sample and transmission schedule as well as providing message status information is support.

The register set of the SENT module can be accessed directly by the CPU for configuration, data read out and status query.

The SENT IP-module supports the following features:

Features

- Conformance with SENT protocol specification J2716 JAN2010
- Data rates of up to 65,8 kbit/s at 3 µs tick length and 6 data nibbles on each channel
- Support of standard tick times (3 µs through 90 µs) and
- Message tick time programmable between 0.2 µs and 90 µs
- 4 SENT channels working independently in parallel
- Status nibble optionally included in the checksum (default not included)
- Sticky interrupt flags, error interrupt optional (default disabled)
- Configurable frame length (default is 24 bit), max data size is 32 bits
- Serial data processing optional (default: disabled)
- Option for bigger frame lengths (must still be fix for each application)
- transparent mode (nibble CRCs are written to the receive control register for SW processing)
- Support of SPC
- Support of trailing Pause Nibble of any length (even longer than 70 ticks)
- Indication of system status: STOP, INITIALIZED, RUNNING, SYNCHRONIZED
- The receiver module will monitor the message for the following error conditions:
  - Calibration pulse length deviates more than +/-25% from the nominal 56 ticks
  - Too many or too few nibbles between calibration pulses.
  - Checksum error.
  - Successive calibration pulse differ by more than 1.5625%
  - Any nibble data values measured as < 0 or >15.
- When any of those errors is detected, the receiver module shall declare that a message error has occurred and ignore the entire message.
- Any of those errors shall cause the receiver to begin searching for a valid calibration pulse to re synchronize.
- Option to enable/disable the check of the next calibration pulse before validation of received data
- Digital Glitch filter suppressing noise
- Buffer overrun detection
- Optional output inversion for use of external open drain transistor
- Optional input inversion for use of external open transistor for level shifting
- Interrupt on status nibble violation
- Programmable Nibble sorting to support LSN or HSN first and relief CPU
- Time stamp generation
- Watch Dog on incoming frames

22.1.2 General Operation

The Single Edge Nibble Transmission encoding scheme (SENT) is intended for use in applications where high resolution sensor data needs to be communicated from a sensor to an Engine Control Unit (ECU). It is intended as a replacement for the lower resolution methods of 10 bit A/D converters and PWM and as a simpler low cost alternative to CAN or LIN. The implementation assumes that the sensor is a smart sensor containing a microprocessor or dedicated logic device (ASIC) to create the signal.

Figure 22-3 shows a typical TC21x/TC22x/TC23x application in which a SENT interface reads a sensor device.

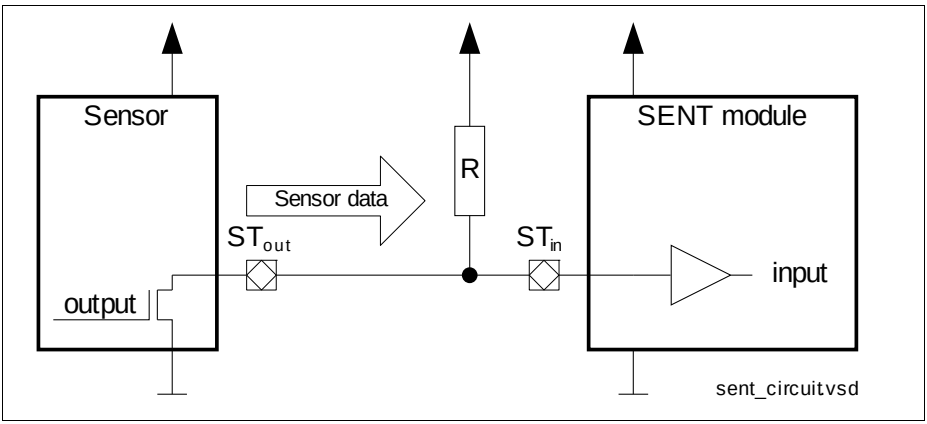


Figure 22-2 SENT to External Device Connection

SENT communication is unidirectional from sensor to controller without any synchronization. The sensor signal is transmitted as a series of PWM blocks measured as falling to falling edge times.

The Short PWM Code (SPC) extension overcomes the drawback of unidirectionality as said above while keeping conformance to the standard.

Figure 22-3 shows a typical TC21x/TC22x/TC23x application in which an SPC enabled SENT ECU reads an SPC enabled SENT sensor device.

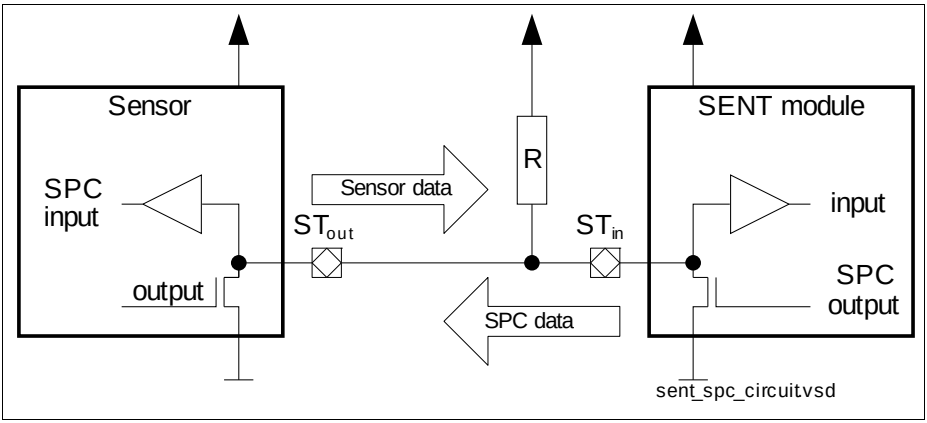


Figure 22-3 SENT SPC to External Device Connection

Some applications are:

- Read out of the external throttle position sensor (e.g. SENT enabled linear hall sensor)
- Receiving pedal position
- Synchronize sampling and read out of up to four sensors on one single line (SPC)
- Selection of 1 out of 4 sensors connected to a single SENT channel.
- Serial connections of the TC21x/TC22x/TC23x to other peripheral devices

22.1.2.1 Definitions

SENT: Single Edge Nibble Transmission

Nibble: Four Bit value between 0 and 15 = half a Byte = one character in hex (0 to F)

SPC: Short PWM Code

PWM: Pulse Width Modulation

ASIC: Application Specific Integrated Circuit

CAN: Controller Area Network

LIN: Local Interconnect Network

ISO: International Organization for Standardization

ECU: Electronic Control Unit

FSM: Finite State Machine

22.1.3 Standard SENT Operation

Standard SENT Mode supports communication fully compliant to J2716 JAN2010, in which only the external device is sending and the ECU is receiving. In this unidirectional communication, both transmitter and receiver use the same data frame format and have the same baud rate. These settings are defined at system integration time and do not change for a given application. Data is received on pin ST. **Figure 22-4** shows the block diagram of the SENT Module when operating in Standard SENT Mode.

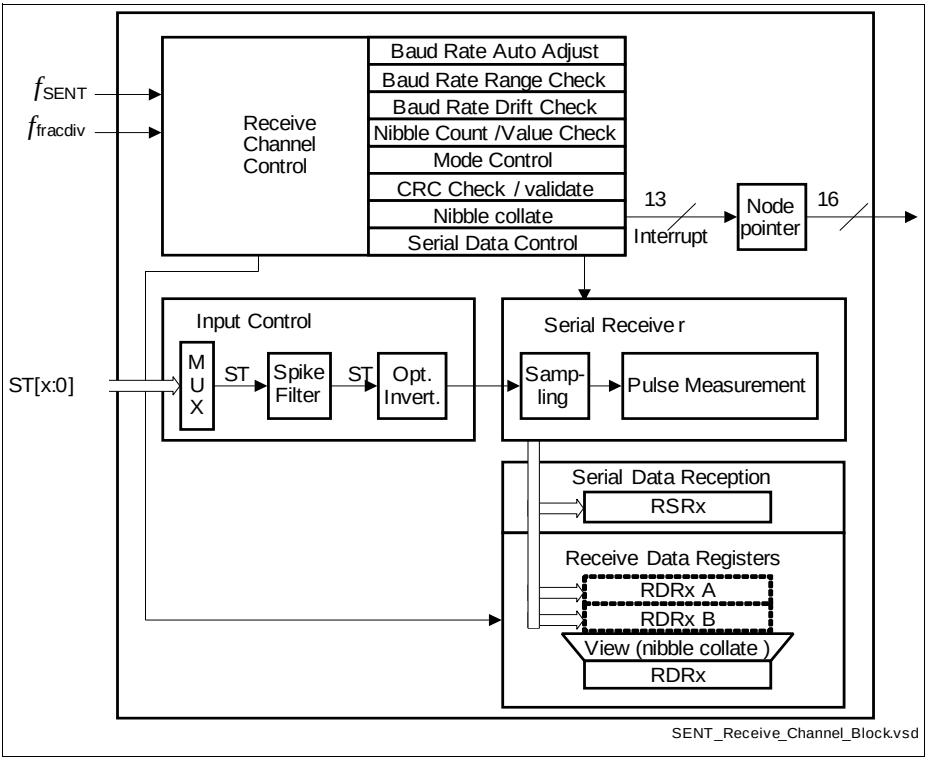


Figure 22-4 Standard SENT Mode Operation

22.1.3.1 Frame Formats and Definitions

This section describes the frame formats and definitions of the SENT protocol.

Basic Definitions

Figure 22-5 shows the layout and definitions of a standard SENT frame. Note that the SENT standard does not specify whether the most significant nibble or the least significant nibble is sent out first.

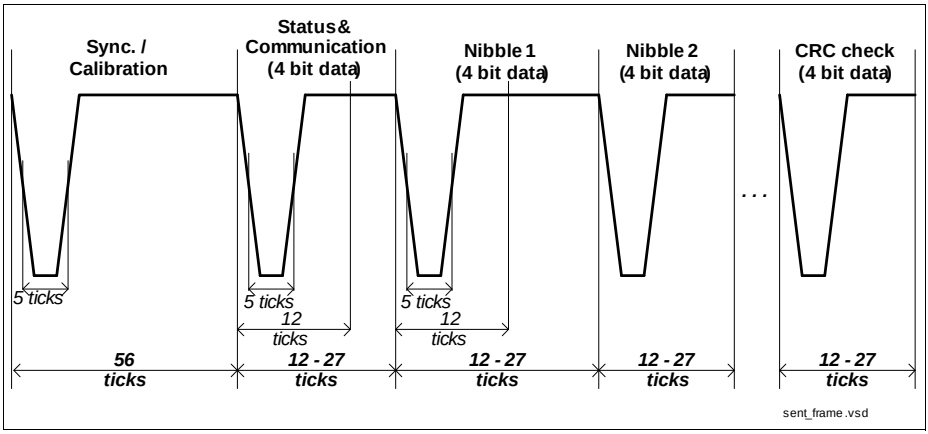


Figure 22-5 Standard Encoding Frame

Serial Data Encoding in Status and Communication Nibble

The Status and Communication nibble is used by sensors that transmit extra serial data such as part numbers or diagnostic information. Data bits from this nibble are constructed across multiple SENT messages to form a Short Serial Message or an Enhanced Serial Message. These messages are alternately referred to as “Slow Channel” data in the J2716 specification.

Table 22-1 Short Serial Data Encoding

Bit Number	Bit Function
0	Reserved for specific application
1	Reserved for specific application
2	Serial Data message bits
3	Message start = 1, otherwise = 0

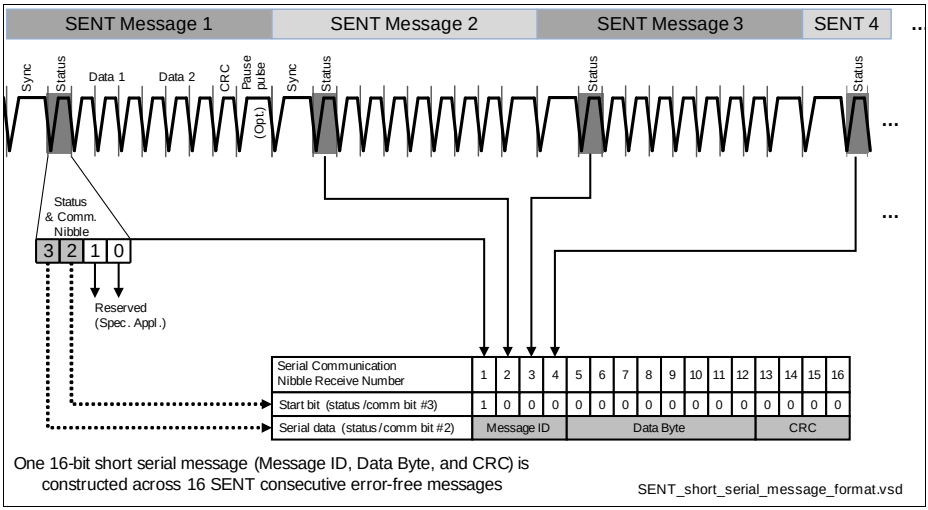


Figure 22-6 Short Serial Message, Serial Data Encoding over 16 messages

Short Serial Message Format

The SENT kernel supports the Short Serial Message Format defined in the J2716 specification. The Short Serial Message Format provides 16 bits of data, constructed across 16 SENT messages. Each of the 16 messages contains one bit of Short Serial Message data in Status and Communication nibble bit 2.

Serial data is communicated in a 16-bit sequence as shown in **Table 22-2**. The start of a Short Serial Message is indicated by a 1 in bit 3 of the Status and Communication nibble. That SENT message and the next 15 must be successfully received (no errors) for the Short Serial Message data to be received. The CRC generation is identical to the CRC generation on the data nibbles. Short Serial Message data is provided in the SDSx register.

Table 22-2 Short Serial Message Format

Serial Communication Nibble Receive No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Startbit (bit # 3)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Serial Data (bit # 2)	Message ID				Data Byte								CRC			

CRC Calculation

SENT signal data and Short Serial Message data is secured with a 4 bit CRC. The standard CRC calculation method is defined in the J2716 JAN2010 specification.

RCR.CRZ defines, if a zero nibble is added for calculation or not.

The alternative checksum nibble is a 4-bit CRC of the data nibbles (including the status nibble if RCR.SNI is set, as used in sensor TLE4998). The CRC is calculated using a polynomial x^4 +x^3 + x^2 + 1 with a seed value of 0101.

In order to facilitate CRC implementations and to avoid ambiguities, an example implementation of the alternative 4-bit CRC is given below. This is used e.g. in the sensor TLE4998 for the signal data. See **Figure 22-7** for details.

```
// Fast way for any µC with low memory and compute capabilities
// contains input data (status nibble, 6 data nibble, CRC)
char Data[8] = {…};
// required variables and LUT
char CheckSum, i;
char CrcLookup[16] = {0, 13, 7, 10, 14, 3, 9, 4, 1, 12, 6, 11, 15,
2, 8, 5};
CheckSum= 5; // initialize checksum with seed “0101”
for (i=0; i<7; i++) {
     CheckSum = CheckSum ^ Data[i];
     CheckSum = CrcLookup[CheckSum];
};
// finally check if Data[7] is equal to CheckSum
```

Note: For the “Enhanced Serial Message Frame Format”, an own 6-bit CRC calculation method is defined by the standard.

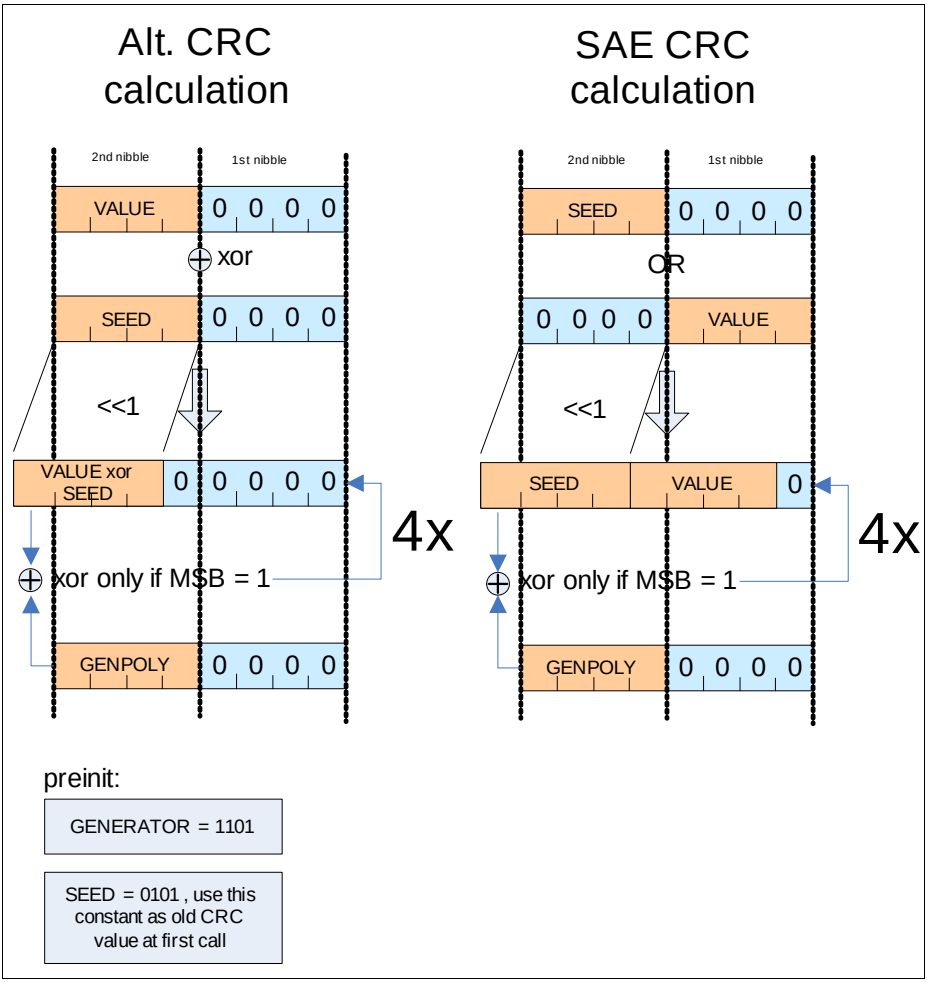


Figure 22-7 Alternate vs. SAE CRC Calculation

Enhanced Serial Message Format

The SENT kernel supports the Enhanced Serial Message Format defined in the J2716 specification. The Enhanced Serial Message Format provides 21 bits of message data and a 6-bit CRC, constructed across 18 SENT messages.The serial data is transmitted bit wise per frame in bits [3:2] of the Status and Communication nibbles of 18 consecutive messages from the transmitter.

Table 22-3 Enhanced Serial Data Encoding

Bit Number	Bit Function
0	Reserved for specific application
1	Reserved for specific application
2	Serial Data message bits
3	Message start = **1** 1 1 1 1 1 0

Enhanced Serial Message Frame

Serial data is communicated in an 18 frame sequence as shown in **Table 22-4**. The start of an Enhanced Serial Message is indicated by the unique pattern “1111110” in bit #3 of the status and communication nibble, constructed across 7 SENT messages. In message 1 - 6 they are set to “1”. In message 7, 13 and 18 they are set to “0”. 18 consecutive SENT messages must be successfully received (no errors) for the Enhanced Serial Message data to be received. The 6-bit Enhanced Serial Message CRC is different from the 4-bit CRC on SENT (fast channel) messages. (See SENT Standard)

An Enhanced Serial Message contains 21 bits of message data and a 6-bit frame-check sequence. A configuration bit (serial data bit #3 of the 8th SENT message's Serial Communication Nibble) determines how the SENT module interprets the serial data.

- Configuration bit = 0: 12-bit data and 8-bit message ID, see **Table 22-5**
- Configuration bit = 1: 16-bit data and 4-bit message ID, see **Table 22-6**

Enhanced Serial Message data is provided in the SDSx register.

Table 22-4 Enhanced Serial Data Frame

Serial Communication Nibble Receive No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Serial Data (bit # 3)	1	1	1	1	1	1	0	C	8-bit ID (7-4)				0	8-bit ID (3-0) or 4-bit data				0
Serial Data (bit # 2)	6-bit CRC						12-bit data field											

Note:

11. *Message 8 Status Nibble bit # 3 is Configuration Bit C that defines the Enhanced Serial Data format:*
    *0: 12-bit data, 8-bit ID, see **Table 22-5***
    *1: 16-bit data, 4-bit ID, see **Table 22-6***
12. *Messages 17-14 Status Nibble bit # 3 contain either the lower 4 bits (Bit 3-0) of 8bit ID or 4 additional data bits.*

Table 22-5 shows in detail the frame structure for Enhanced Serial Data Frames with Configuration Bit = 0.

Table 22-5 Configuration Bit = 0

Serial Communication Nibble Receive No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Serial Data (bit # 3)	1	1	1	1	1	1	0	0	8-bit ID (7-4)				0	8-bit ID (3-0)				0
Serial Data (bit # 2)	6-bit CRC						12-bit data field											

Table 22-6 shows in detail the frame structure for Enhanced Serial Data Frames with Configuration Bit = 1.

Table 22-6 Configuration Bit = 1

Serial Communication Nibble Receive No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Serial Data (bit # 3)	1	1	1	1	1	1	0	1	4-bit ID (3-0)				0	4-bit data (15-12)				0
Serial Data (bit # 2)	6-bit CRC						12-bit data field											

22.1.4 SPC Operation

The module supports a SPC (Short PWM Code) protocol, which enhances the standardized SENT protocol (Single Edge Nibble Transmission) defined by J2716 JAN2010. SPC enables the use of enhanced protocol functionality due to the ability to select between “synchronous”, “range selection” and “ID selection” protocol mode or even “bidirectional transmit mode”.

22.1.4.1 Synchronous Transmission

In the “synchronous” mode, the sensor (slave) starts to transfer a complete data frame only after a low pulse is forced by the master on the OUT pin. This means that the data line is bidirectional - an open drain output of the micro controller (master) sends the trigger pulse. The sensor then initiates a sync pulse and starts to calculate the new output data value. After the synchronization period, the data follows in form of a standard SENT frame, starting with the status, data and CRC nibbles. At the end, an end pulse allows the CRC nibble decoding and indicates that the data line is idle again. The timing diagram in **Figure 22-8** visualizes a synchronous transmission

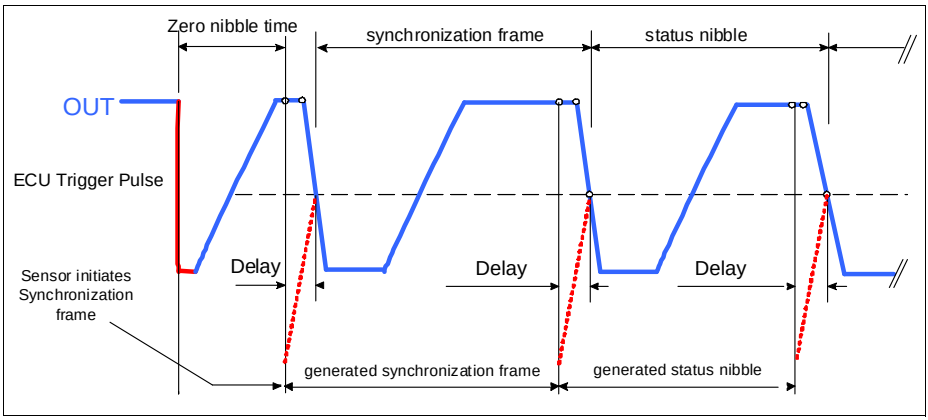


Figure 22-8 Synchronous Transmission (SPC)

22.1.4.2 Range Selection

The low time duration of the master can be used to select the range of the sensor in SPC dynamic range selection mode.

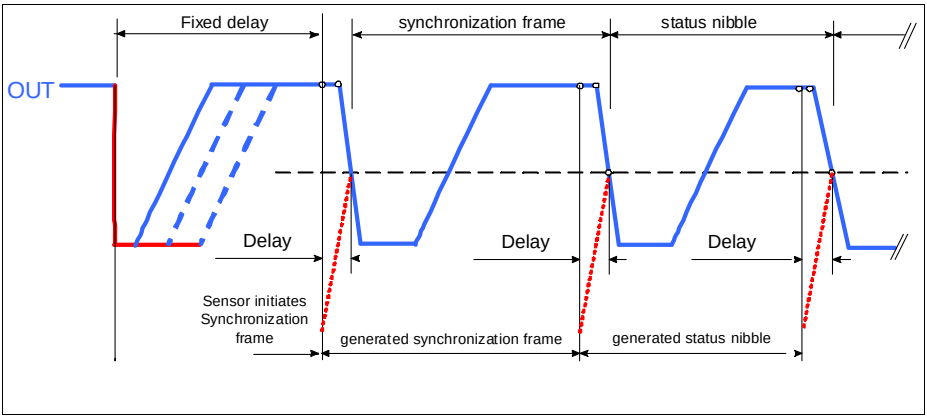


Figure 22-9 Range (SPC)

22.1.4.3 ID Selection

This functionality is similar to the previous mode, but instead of switching the range of one sensor, one of up to four sensors are selectable on a bus (bus mode, 1 master with up to 4 slaves). This allows parallel connection of up to 4 sensors using only three lines (VDD, GND, OUT), as illustrated in **Figure 22-10**.

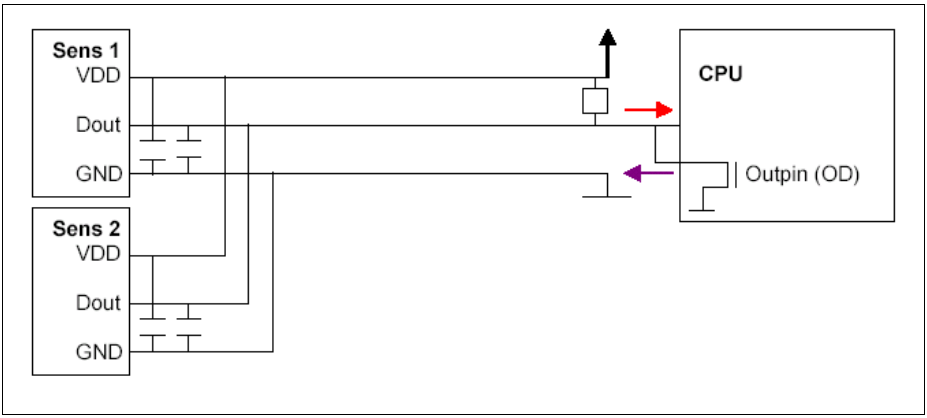


Figure 22-10 ID Selection (SPC)

22.1.4.4 Bidirectional Transmit Mode

After addressing the sensor the ECU interrupts the sensor sync pulse by pulling down the line. The ECU starts to transmit own nibbles on the time base of the selected sensor. The ECU has to adapt the sensors timing from an earlier received cycle.

The sensor detects the falling edge that infringes the protocol and switches to receive mode as illustrated in **Figure 22-11**.

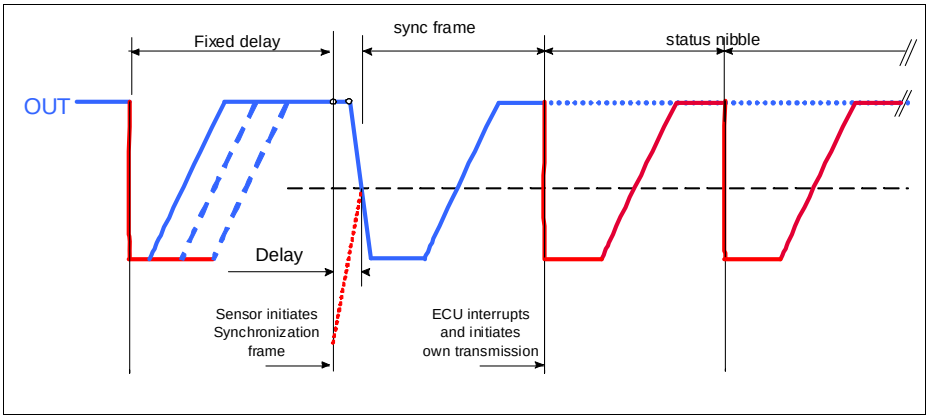


Figure 22-11 Bidirectional Transmit Mode (SPC)

22.1.4.5 SPC Timing

An SPC transmission is initiated by a Master pulse on the OUT pin. To detect a low level on the OUT pin, the voltage must be below a threshold Vthf. The sensor detects that the OUT line has been released as soon as Vthr is crossed. **Figure 22-12** shows the timing definitions for the master pulse. The master low time tmlow as well as the total trigger time tmtr are individual for the different SPC modes and are given in the sensors specifications.It is recommended to chose the typical master low time exactly between the minimum and the maximum possible time given by the connected sensor: tmlow,typ = (tmlow,min + tmlow,max) / 2.

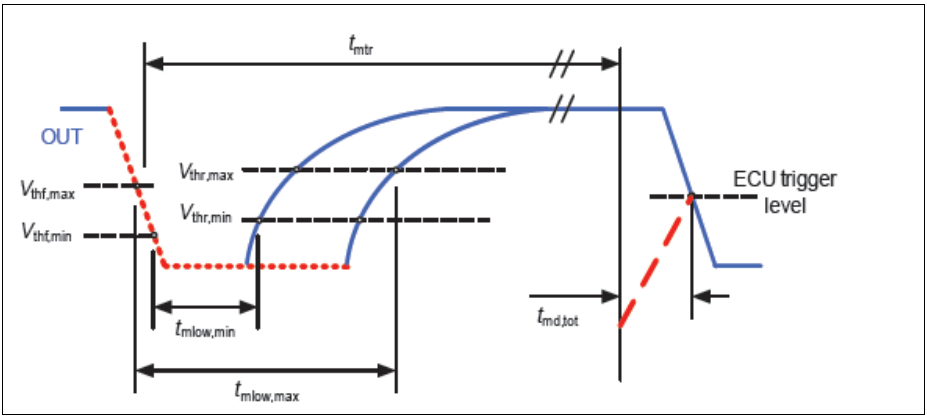


Figure 22-12 Timing (SPC)

22.1.4.6 Abort of Frames

Only a reset condition of the device, a hard suspend or choosing Mode 0 by clearing bit field SCRx.TRIG can abort a current SPC transmission immediately. During hard suspend, the port output drives its current value until suspend is terminated. If the receive channel becomes disabled, the soft suspend mode is requested, or the sleep mode is entered, the SENT module still does start a new frame transmission. If one of these three conditions becomes active during a running frame transmission, the frame transmission is completely finished before the requested abort state is entered. Note that in this case no frame finished interrupt is generated any more.

Note: Received frames are aborted immediately on a disable, suspend or sleep request.

22.1.5 Baud Rate Generation

The nominal baud rate of each channel is individually adjustable. This is required as it is depending from the connected sensor and its current deviation from nominal frequency.

The actual baud rate of the channel follows automatically the baud rate of the connected device and its current deviation from nominal frequency. The adaptation range is +- 25% as specified in J2716 JAN2010.

Chapter 22.1.5 shows in detail how the baud rate for each channel is adjusted. 1 / f_{tick_x} defines the resulting tick length.

In the first stage, a global fractional divider serves as pre divider. Its intermediate frequency $f_{fracdiv}$ can be set up so that it is handy as input frequency for the different SENT channels. The clock signal f_{pdiv} of a channel must always be at least 20 times the nominal tick frequency of the channel. This is to allow for the highly flexible and big tolerance of +-25% versus the sending device. In addition the module must be able to detect a deviation in the length of 2 consecutive Synchronization / Calibration pulses of 1.5625% (1/64) and adapt the own baud rate. The tick time is typically 3 µs but can vary by standard and take values up to 10 µs. Future application might require even shorter tick times for higher repetition rates. This is why this implementation allows for an even extended range of 0.2 … 90 µs. To achieve this each channel has its own fractional divider. This allows to downscale the frequency of f_{tick} precisely to the required tick frequency. For details on the principles of a fractional divider, please refer to the SCU chapter of TC21x/TC22x/TC23x section “Clock Control”.

The SENT module provides two clock signals to the channels (**Figure 22-13**):

- f_{SENT}
  This is the module clock that is used inside the SENT kernel for control purposes such as clocking of control logic and register operations. The frequency of f_{SENT} is always identical to the system clock frequency f_{SYS}. The clock control register SENT_CLC makes it possible to enable/disable f_{SENT} under certain conditions.
- $f_{fracdiv}$
  This clock is the module clock that is used inside the SENT kernel for baud rate generation of the serial channels. The fractional divider register SENT_FDR controls the frequency of $f_{fracdiv}$. Usually not required and set to 1.

The channels generate two local clock signals:

- f_{pdiv_x}
  This clock is the channel clock that is used inside the SENT channel x for baud rate generation of the serial channel. The divider register SENT_CPDRx controls the frequency of f_{pdiv_x}.
- f_{tick_x}
  This clock is the channel clock that is used inside the SENT channel as the baud rate frequency. The fractional divider register SENT_CFDRx controls the frequency of

f_{tick_x}.The higher the value of DIV the higher the precision with which the Synchronization / Calibration pulse and the deviation in the length of 2 consecutive Synchronization / Calibration pulses (drift) can be measured.
DIV must be set in an interval of [560, 3276].

Each time a Synchronization / Calibration pulse starts DIVM is measured. DIVM is the result of measuring the calibration pulse duration with f_{pdiv_x}. At the end of the Synchronization / Calibration pulse this value is taken as new divider of the CFDR.

Each time a pulse starts, the internal accumulator of the CFDR is preset with DIVM / 2. This is required to center the data eye and to make the margin symmetrical.

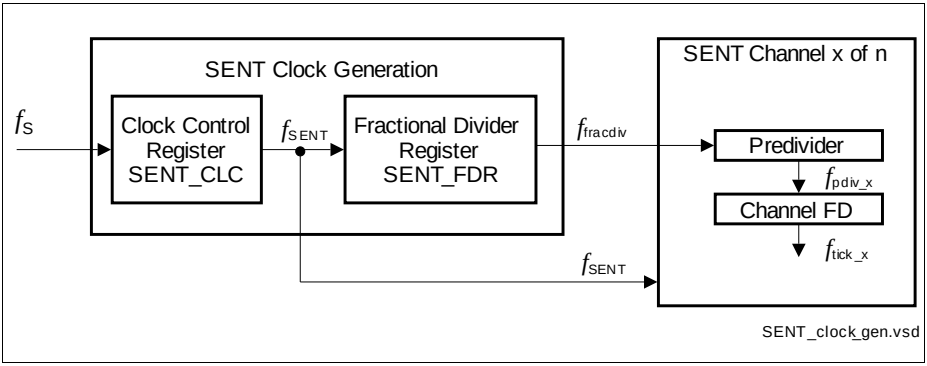


Figure 22-13 SENT Module Clock Generation

The following two formulas define the frequency of $f_{fracdiv}$:

$f_{fracdiv} = f_{SENT}$ / (1024 - SENT_FDR.STEP); FDR.DM = 01_B (22.1)

$f_{fracdiv} = f_{SENT}$ x SENT_FDR.STEP / 1024 with STEP = 0 … 1023; FDR.DM = 10_B (22.2)

The following formula defines the frequency of f_{pdiv_x}:

$f_{pdiv_x} = f_{fracdiv}$ / (SENT_CPDRx.PDIV + 1) (22.3)

The following formula defines the nominal frequency of f_{tick_x}. For the actual frequency of f_{tick_x} DIV is replaced by DIVM after the current sensor frequency was validly measured.

$f_{tick_x} = f_{pdiv_x}$ x 56 / SENT_CFDRx.DIV with DIV = 560 … 3276 (22.4)

22.1.6 Error Detection Capabilities

Each SENT channel can detect and signal the following error conditions:

Protocol Level:

- Calibration pulse length deviates more than +/-25% from the nominal 56 ticks
- Too many or too few nibbles between calibration pulses
- Checksum error
- Successive calibration pulse differ by more than 1.5625%
- Any nibble data values measured as < 0 or >15
- Wrong Status and Communication nibbles
- Serial Communication CRC error

Transfer Management Level:

- Receive Data Buffer Overrun
- SPC Data Buffer Underrun

22.1.7 Digital Glitch Filter

Very slow slopes and signal noise can lead to fast transitions of the input signal. These unwanted transitions are suppressed by a digital glitch filter similar to a Filter and Prescaler Cell (FPC) in Delayed Debounce Filter Mode with up and down (no reset).

It is built for filtering very fast transitions only. The filter calculates the integral of the signal. If the integral reaches a programmable saturation point, the signal change is notified to the pulse measurement unit. Thus it helps to find the exact pulse length for said slow slopes in noisy environment.

The glitch filter is clocked with f_{pdiv}. If the state of the input sample differs from the current output signal value, the internal counter is incremented by one. When the state of the input sample matches the current output signal value and the timer is not in idle, the timer is decremented by one. When the timer matches the compare value stored in IOCRx.DEPTH, the level of the output signal line is inverted. The timer will be reset and set to idle state again.

The depth of the filter can be programmed to a value between 1 and 15. Typically a depth of 3 to 5 T_{pdiv} is sufficient. By default it is cleared. If IOCRx.DEPTH is cleared the filter is inactive. Nevertheless, the input signal is sampled with f_{pdiv}.

The internal signal after the filter will change value not before the new value was sampled DEPTH times. If during this period a spike occurs, it takes 2 x T_{pdiv} times longer for the internal signal to change value. The filters principal implies a delay of the signal by (DEPTH x T_{pdiv}).

Upon detection of glitch during rising or falling edge, IOCRx.REG or IOCRx.FEG is set. The rising / falling edge glitch flags must be reset by software.

Figure 22-14 shows the digital glitch filter:

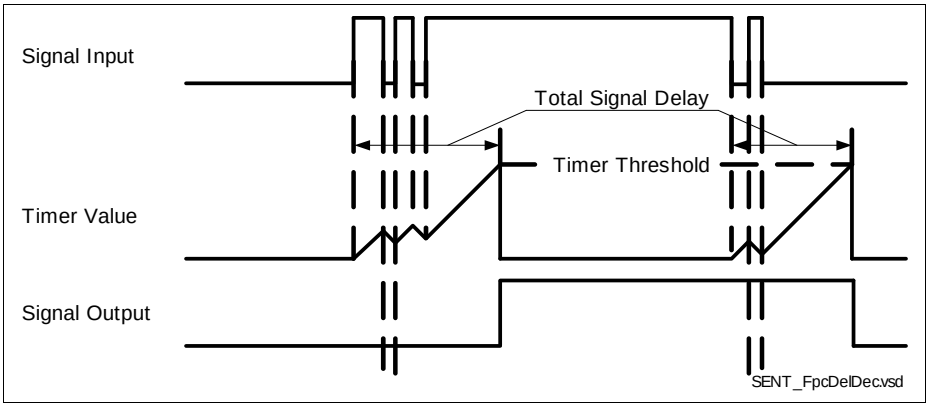


Figure 22-14 Digital Glitch Filter

22.1.8 Interrupts

4 Interrupt request lines are available for the SENT module.

RDI indicates a receive data interrupt. It is activated when a received frame is moved to a Receive Data Register RDR. RSI indicates a receive frame success interrupt, i.e. the CRC was successful. Both RDI and RSI will be issued together in normal use cases where the frame size is not bigger than 8 nibbles and CRC is correct. RBI indicates a receive buffer overrun interrupt. It is activated when a new frame is transferred to a Receive Data Register RDR while the old value was still not read by the host (“overwrite”), i.e. the kernel wants to set any of the two interrupts RSI and RDI and finds any of these two interrupts already set. TDI indicates a transmit interrupt. It is activated when data is moved from a SCR to a transmit shift register. TBI indicates a transfer buffer under run interrupt. It is set after data has been completely transferred (PLEN exceeded) and no new data was written to SCRx. In addition the protocol error interrupts are available: FRI, FDI, NNI, NVI, CRCI. If one of the protocol interrupts is activated, data is to be treated as invalid according to J2716 JAN2010. WSI, SDI SCRI treat the interrupts referring to the Status and Communication nibble. WDI is the Watch Dog Error Interrupt. It is issued if the time between two frames is too long.

For acceleration of the interrupt service routine, a Register INTOV is implemented that holds a flag for each channel. This flag is automatically set if there is an interrupt pending for the channel which is enabled. It is automatically reset, if no more enabled interrupt is pending for this channel.

The interrupt request or the corresponding interrupt set bit (in register INTSET) can trigger the interrupt generation at the selected interrupt node. The service request pulse is generated independently from the interrupt flag in register INTSTATx. The interrupt flag can be cleared by software by writing to the corresponding bit in register INTCLR.
If more than one interrupt source is connected to the same interrupt node pointer (in register INPx), the requests are combined to one common line.

22.1.9 Trigger Outputs

Any interrupt source can be used as trigger output TRIGO outside the module. Each TRIGO is connected to a IR input. See **Table 22-9 “Service Request Lines of SENT” on Page 22-72**.

22.2 SENT Kernel Registers

This section describes the kernel registers of the SENT module. All SENT kernel register names described in this section will be referenced in other parts of the TC21x/TC22x/TC23x User's Manual by the module name prefix "SENT_" for the SENT interface.

All registers in the SENT address spaces are reset with the application reset (definition see SCU section "Reset Operation").

SENT Kernel Register Overview

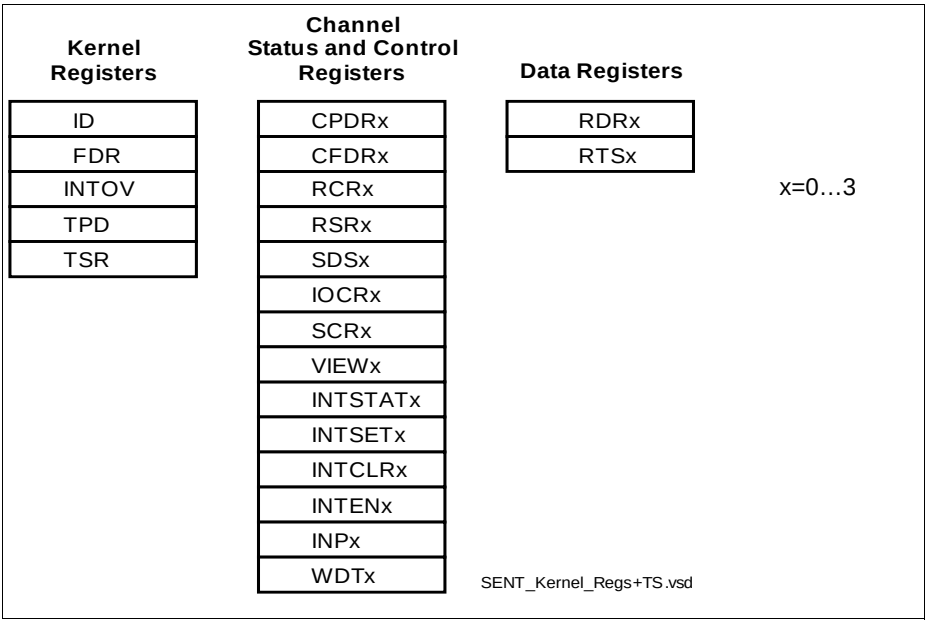


Figure 22-15 SENT Kernel Registers

The complete and detailed address map of the SENT module is described in **Table 22-7** on **Page 22-23**.

Note: x can take the values 0 … 3

Table 22-7 Registers Address Space - SENT Kernel Registers

Module	Base Address	End Address	Note
SENT	$F0003000_H$	$F0003AFF_H$	–

Table 22-8 Registers Overview - SENT Kernel Registers

Register Short Name	Register Long Name	Offset Address [1]	Access Mode		Description see
			Read	Write	
SENT_CLC	SENT Clock Control Register	00_H	SV, U	SV, E, P	Page 22-77
-	reserved	04_H	BE	BE	
SENT_ID	Module Identification Register	08_H	SV, U	BE	Page 22-27
SENT_FDR	Module Fractional Divider	$0C_H$	SV, U	E, P	Page 22-28
-	reserved	10_H	BE	BE	
SENT_INTOV	Interrupt Overview Register	14_H	SV, U	BE	Page 22-54
SENT_TSR	Module Time Stamp Register	18_H	SV, U	BE	Page 22-29
SENT_TPD	Module Time Stamp Predivider Register	$1C_H$	SV, U	E, P	Page 22-30
-	reserved	20_H - $7C_H$	BE	BE	
SENT_RDRx	Receive Data Register x	80_H - 80_H+x*4_H	SV, U	BE	Page 22-47
-	reserved	84_H+3*4 $_H$ - $E4_H$	BE	BE	
OCS	OCDS Control and Status Register	$E8_H$	U, SV	SV, E, P	Page 22-78
ACCEN0	Access Enable Register 0	FC_H	U, SV	SV, SE, P	Page 22-79
ACCEN1	Access Enable Register 1	$F8_H$	U, SV	SV, SE, P	Page 22-80
KRST0	Reset Control Register 0	$F4_H$	U, SV	SV, E, P	Page 22-81
KRST1	Reset Control Register 1	$F0_H$	U, SV	SV, E, P	Page 22-82
KRSTCLR	Reset Status Clear Register	EC_H	U, SV	SV, E, P	Page 22-83

Table 22-8 Registers Overview - SENT Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Address [1]	Access Mode		Description see
			Read	Write	
SENT_CPDRx	Channel Pre Divider Register x	100_H + x * 40_H	SV, U	SV, U, P	Page 22-31
SENT_CFDRx	Channel Fractional Divider Register x	104_H + x * 40_H	SV, U	SV, U, P	Page 22-32
SENT_RCRx	Receiver Control Register x	108_H + x * 40_H	SV, U	SV, U, P	Page 22-33
SENT_RSRx	Receive Status Register x	$10C_H$ + x * 40_H	SV, U	BE	Page 22-41
SENT_SDSx	Serial Data and Status Register x	110_H + x * 40_H	SV, U	BE	Page 22-42
SENT_IOCRx	Input and Output Control Register x	114_H + x * 40_H	SV, U	SV, U, P	Page 22-43
SENT_SCRx	SPC Control Register x	118_H + x * 40_H	SV, U	SV, U, P	Page 22-52
SENT_VIEWx	Receive Data View Register x	$11C_H$ + x * 40_H	SV, U	SV, U, P	Page 22-49
SENT_INTSTATx	Interrupt Status Register x	120_H + x * 40_H	SV, U	BE	Page 22-55
SENT_INTSETx	Interrupt Set Register x	124_H + x * 40_H	SV, U	SV, U, P	Page 22-61
SENT_INTCLRx	Interrupt Clear Register x	128_H + x * 40_H	SV, U	SV, U, P	Page 22-63
SENT_INTENx	Interrupt Enable Register x	$12C_H$ + x * 40_H	SV, U	SV, U, P	Page 22-65
SENT_INPx	Interrupt Node Pointer Register x	130_H + x * 40_H	SV, U	SV, U, P	Page 22-68
SENT_WDTx	Watch Dog Timer Register x	134_H + x * 40_H	SV, U	SV, U, P	Page 22-40
-	reserved	138_H + x * 40_H - $13F_H$ + x * 40_H	BE	BE	

Table 22-8 Registers Overview - SENT Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Address [1]	Access Mode		Description see
			Read	Write	
-	reserved	$9F0_H$ - $A7C_H$	BE	BE	
SENT_RTSx	Receive Time Stamp x	A80 + x * 4_H	SV, U	BE	Page 22-48
	Reserved	$AA8_H$ - AFC_H	BE	BE	-

1) The absolute register address is calculated as follows:
Module Base Address (Table 22-7) + Offset Address (shown in this column)

List of Access Protection Abbreviations

U - User Mode

SV - Supervisor Mode

BE - Bus Error

nBE - no Bus Error

P - Access Protection, as defined by the ACCEN Register

E - ENDINIT

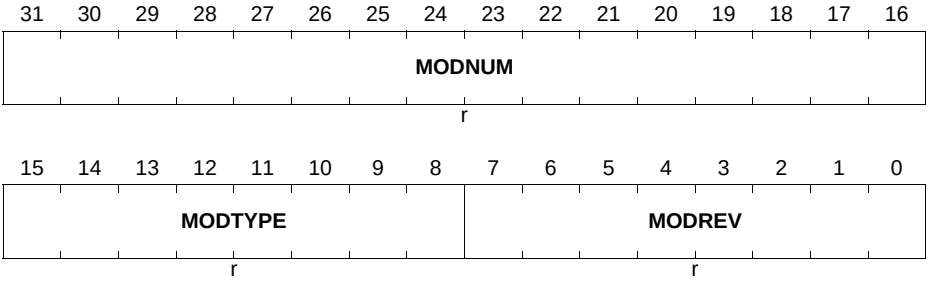
SE - Safety ENDINIT

22.2.1 Module Control

Module Identification Register

The SENT Module Identification Register ID contains read-only information about the module version.

ID
Module Identification Register (08$_H$) Reset Value: 0080 C0XX$_H$

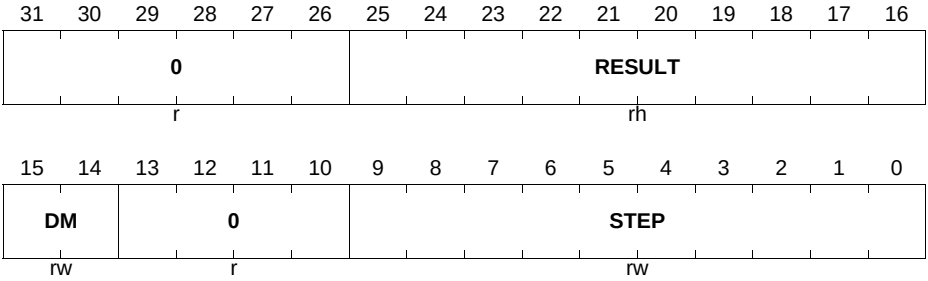


Field	Bits	Type	Description
MODREV	[7:0]	r	**Module Revision Number** This bit field defines the module revision number. The value of a module revision starts with 01$_H$ (first revision).
MODTYPE	[15:8]	r	**Module Type** This bit field defines the module as a 32-bit module: C0$_H$
MODNUM	[31:16]	r	**Module Number Value** This bit field defines the module identification number for the SENT: 0080$_H$

Fractional Divider Register

The Fractional Divider Register controls the input clock $f_{fracdiv}$ of all SENT channels.

SENT_FDR
SENT Fractional Divider Register ($0C_H$) Reset Value: 0000 0000$_H$

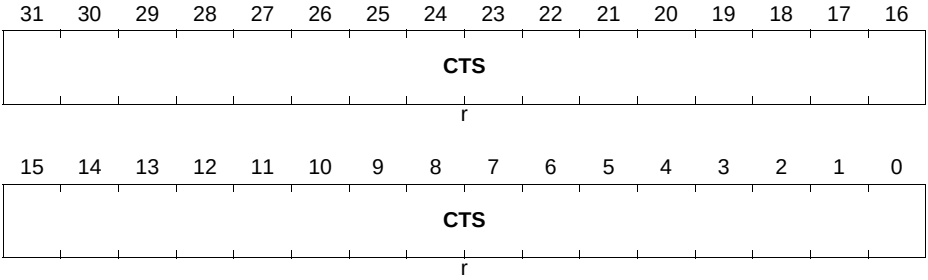


Field	Bits	Type	Description
STEP	[9:0]	rw	**Step Value** Reload or addition value for RESULT.
DM	[15:14]	rw	**Divider Mode** DM selects normal or fractional divider mode. 00_B Fractional divider is switched off; no output clock is generated. The Reset External Divider signal is 1. RESULT is not updated (default after System Reset). 01_B Normal Divider Mode selected. 10_B Fractional Divider Mode selected. 11_B Fractional divider is switched off; no output clock is generated. RESULT is not updated.
RESULT	[25:16]	rh	**Result Value** Bit field for the addition result.
0	[13:10], [31:26]	r	**Reserved** Read as 0; should be written with 0.

Module Time Stamp Register

The SENT Module Time Stamp Register contains read-only information about the current time given in clock cycles generated from TPD since last reset while module was clocked. Writing to TPD clears this register.

TSR
Time Stamp Register **(18_H)** **Reset Value: $0000\ 0000_H$**

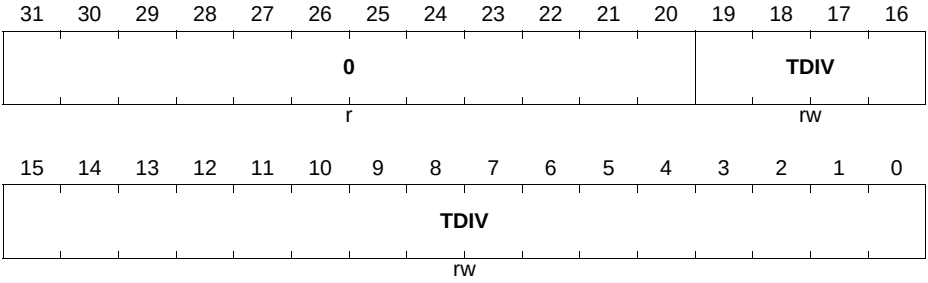


Field	Bits	Type	Description
CTS	[31:0]	r	**Current Time Stamp for the Module** This bit field shows the current time stamp.

Module Time Stamp Pre Divider Register

The SENT Module Time Stamp Predivider Register contains the pre divider that defines the time resolution of TSR. Writing to TPD clears register TSR.

TPD
Time Stamp Predivider Register ($1C_H$) Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
TDIV	[19:0]	rw	**Divider Factor of Pre Divider for TSR** Divides $f_{fracdiv}$ by (TDIV + 1) and drives TSR.
0	[31:20]	r	**Reserved** Read as 0; should be written with 0.

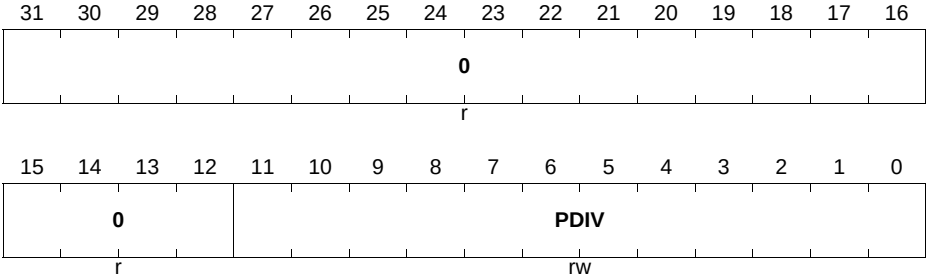
22.2.2 Channel Baud Rate Registers

Channel Pre Divider Register

The Channel Pre Divider Register CPDRx contains the pre divider that is related to the SENT channel baud rate.

See **Equation (22.3)**

CPDRx (x = 0-3)
Channel Pre Divider Register x **(100_H+40_H*x)** **Reset Value: 0000 0000_H**



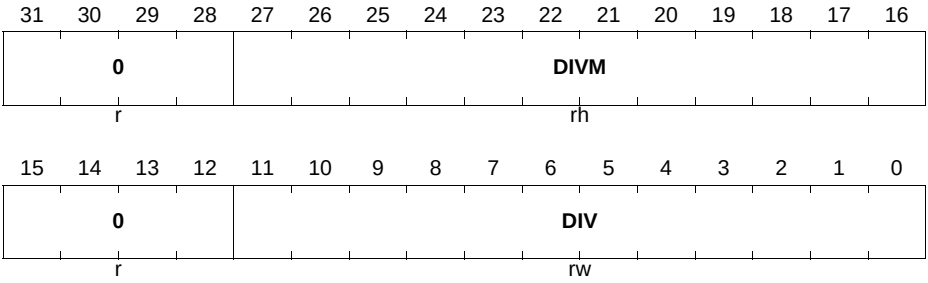
Field	Bits	Type	Description
PDIV	[11:0]	rw	**Divider Factor of Pre Divider for Channel x** Divides $f_{fracdiv}$ by (PDIV + 1) and delivers f_{pdiv_x} to the Channel Fractional Divider. RCR.CEN must be cleared before changing CPDR.PDIV or CFDR.DIV.
0	[31:12]	r	**Reserved** Read as 0; should be written with 0.

Channel Fractional Divider Register

The Channel Fractional Divider Register CFDRx contains control bits/bit fields that are related to the SENT channel baud rate.

See **Equation (22.4)** in **Chapter 22.1.5** for a detailed description.

CFDRx (x = 0-3)
Channel Fractional Divider Register x(104_H+40_H*x) **Reset Value: 0000 0000_H**



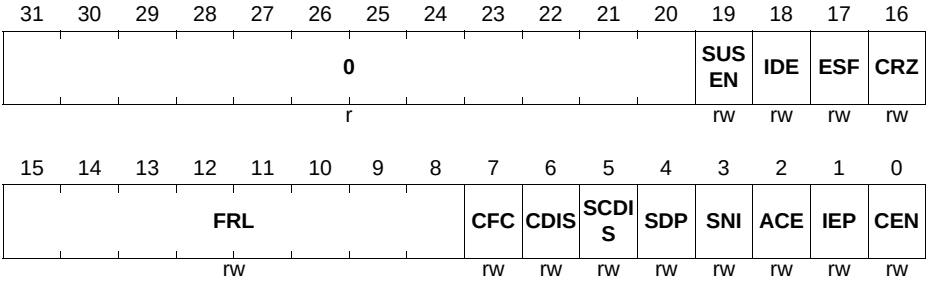
Field	Bits	Type	Description
DIV	[11:0]	rw	**Divider Value** Initial and reference divider value for the CFDR. DIV must be programmed > 0. If cleared, DIV becomes 1. If written, DIVM is updated automatically with the same value.RCR.CEN must be cleared before changing CPDR.PDIV or CFDR.DIV.
DIVM	[27:16]	rh	**Measured Divider Value** DIVM is automatically updated by HW to adjust the receiver frequency to the current sender frequency. This value is kept automatically in the range of 75% DIV < DIVM < 125% DIV Write data is ignored.
0	[15:12], [31:28]	r	**Reserved** Read as 0; should be written with 0.

22.2.3 Receiver Control and Status Registers

Receiver Control Register

The Receiver Control Register RCRx contains control bits/bit fields that are related to the SENT receiver operation.

RCRx (x = 0-3)
Receiver Control Register x **(108_H+40_H*x)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
CEN	0	rw	**Channel Enable** When CEN is set, the receiver of channel x is enabled. The internal receiver state machine can be initialized by switching the channel off and on. This does not change the current register content. 0_B channel x disabled (default) 1_B channel x enabled
IEP	1	rw	**Ignore End Pulse** When IEP is set, an end pulse is ignored. An end pulse can be generated in SPC mode or as pause pulse. 0_B End Pulse not ignored (default) 1_B End Pulse ignored For systems with an end pulse, during synchronize or re-synchronize of reception, if calibration pulses are detected one immediately following the other, the first calibration pulse shall be ignored as it may be a pause pulse with duration matching the calibration pulse range.

Field	Bits	Type	Description
ACE	2	rw	**Alternative CRC Mode Enable** When ACE is set, the CRC is calculated in an alternative way for both: fast (signal) and slow (serial message) data path. *Note: If ESF is set, the standard 6 bit CRC is always used for the serial message and ACE is ignored.* 0_B Serial CRC calculation as specified in J2716 JAN2010 (default) 1_B Alternative: 4 bit in parallel CRC calculation as used e.g. in hall sensor TLE4998C.
SNI	3	rw	**Status Nibble Included in CRC** When SNI is set, the status Nibble is included in (signal data) CRC. 0_B Status Nibble not included in CRC (default) 1_B Status Nibble included in CRC (as used e.g. in hall sensor TLE4998C).
SDP	4	rw	**Serial Data Processing Mode** This bit switches automatic serial data processing on. 0_B Automatic Serial Data Processing is disabled. Status and Communication nibble can be read from RSRx for SW processing. (default) 1_B Automatic Serial Data Processing is enabled. Status and Communication nibble can be read from RSRx; Message ID, Serial Data and SCRC can be read from SDSx after serial data interrupt SDI is activated.
SCDIS	5	rw	**CRC for Serial Data Disabled Mode** This bit selects the CRC disabled mode. 00_B CRC is enabled (default) 01_B CRC is disabled CRC nibble can be read from SDSx. The CPU must perform the CRC on the current data by SW.

Field	Bits	Type	Description
CDIS	6	rw	**CRC Disabled Mode** This bit selects the CRC disabled mode. 00_B CRC is enabled (default) 01_B CRC is disabled CRC nibble can be read from RSRx. The CPU must perform the CRC on the current data by SW.

Field	Bits	Type	Description
CFC	7	rw	**Consecutive Frame Check** This bit determines the way the most recently received frame buffer is indicated as valid. 0_B **Check against Past Sync Pulse** The current Synchronization / Calibration Pulse is compared to the Synchronization / Calibration Pulse received immediately before. The whole frame is invalid if the Synchronization / Calibration Pulse length differs from the length of the Synchronization / Calibration Pulse before by more than 1.5625%. In this case of error, its length is not used as new reference. In case the check passes and no other error occurs the Frame Buffer is indicated valid immediately after CRC calculation result is correct. Resynchronization: On the third successive calibration pulse error, the current calibration pulse value is considered as valid and the message accepted unless the message frame contains other errors. In both cases a receive data interrupt (RDI), or a referring error interrupt is issued. 1_B **Check against Future Sync Pulse** The current Synchronization / Calibration Pulse is compared with the Synchronization / Calibration Pulse received immediately after the current frame. The whole frame is invalid if the Synchronization / Calibration Pulse length differs from the length of the following Synchronization / Calibration Pulse by more than 1.5625%. Resynchronization: In this case of error, the current length is used as new reference. Note: The whole frame can be indicated valid only after additionally measuring the Synchronization / Calibration Pulse of the successive frame.

Field	Bits	Type	Description
FRL	[15:8]	rw	**Frame Length** FRL determines the number of data nibbles per frame that the SENT channel x is setup for. Note that FRL does not include the Synchronization / Calibration Pulse, the Status and Communication nibble, the CRC nibble nor the additional zero length nibble that might be introduced by use of SPC. 00000000_B No data nibble 00000001_B 1 data nibble 00000010_B 2 data nibbles 00000011_B 3 nibbles ... ... 00001000_B 8 nibbles Maximum in normal length mode ... ... 11111111_B 255 nibbles If more than 8 nibbles are configured, please note: In addition to the receive success interrupt RSI at the successfully received end of a frame, a receive data interrupt RDI is issued each time 8 nibbles have been transferred to the Receive Data Register RDRx. At the end of a frame, RDI is issued if RSI is issued. If an error occurred, RDI is not set at the end of a frame. If no CRC has been received at the point in time where RDI is issued, the receive data interrupt is no indication whether or not the transfer was successful so far. A CRC Error Interrupt is issued at the end of the frame if Automatic CRC check is enabled and the CRC is wrong.

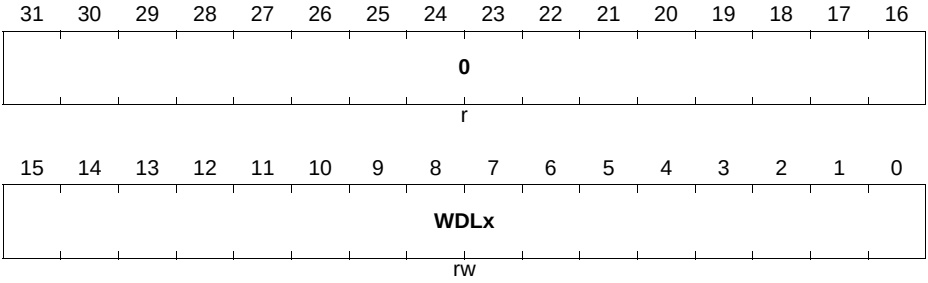
Field	Bits	Type	Description
CRZ	16	rw	**CRC with Zero Nibble for Serial Data** This bit selects the CRC method. If CRZ is cleared, augmentation is selected, (i.e a ZERO NIBBLE is added at the end of CRC calculation (only in calculation)). E.g. as 7th nibble (in case of 6 data nibbles) 00_B Augmentation is selected for both 4-bit message CRC and the 4-bit CRC of the serial messages (legacy, 16 frames) (default) 01_B Augmentation is switched off for both 4-bit message CRC and the 4-bit CRC of the serial messages (legacy, 16 frames). The enhanced serial message (18 Frames, 6-bit CRC) is not controlled by CRZ but the 6-bit CRC is always augmented according to standard.
ESF	17	rw	**Enhanced Serial Frame Mode** This bit selects the serial frame structure. 0_B short (16 frames, 4 bit ID, 8 bit data, 4 bit CRC) 1_B enhanced (18 frames, 4 or 8 bit ID, 12 or 16 bit data, 6 bit CRC)
IDE	18	rw	**Ignore Drift Error Mode** This bit selects if drift errors lead to frame rejection and if an interrupt (INTSTAT.FDI) is generated. Used, if sensors are triggered by SPC. During a long pause period the accumulated drift could be more than 1.5625%. In this special case setting IDE is useful. 0_B Drift Errors enabled (default) 1_B Drift Errors disabled
SUSEN	19	rw	**Suspend Enable** This bit makes it possible to set the SENT channel into Suspend Mode via OCDS (on chip debug support): 0_B An OCDS suspend trigger is ignored by this SENT channel. 1_B An OCDS suspend trigger disables the SENT channel: As soon as the SPC sender logic of the SENT channel becomes idle, the module is stopped while all registers of the channel stay readable. The receiver is stopped unconditionally. A partly received frame is discarded. Bit SUSEN is reset via OCDS Reset.

Field	Bits	Type	Description
0	[31:20]	r	**Reserved** Read as 0; should be written with 0.

Watch Dog Timer Register

The Watch Dog Timer Register WDTx contains the time out value for channel x.
The internal Watch Dog timer is cleared and started automatically each time an RDI (Receive Data Interrupt Request Flag) is set in the INTSTAT of the referring channel and also on any write to WDL that sets WDL>0. The value entered here defines the time in multiples of T_{tick_x} (defined in CFDR) from the last RDI on channel x.
The internal Watch Dog Timer is compared to WDLx.
A match triggers interrupt WDI and stops the internal Watch Dog Counter x.
After such a match, the Watch Dog Timer will be cleared and started automatically with clearing bit WDI (by setting INTCLR.WDI). I.e. the Watch Dog Counter is running only if interrupt flag WDI is cleared.
If WDL is cleared, the WDTx is stopped/disabled.

WDTx (x = 0-3)
Watch Dog Timer Register x **(134_H + x * 40_H)** **Reset Value: 0000 0000_H**

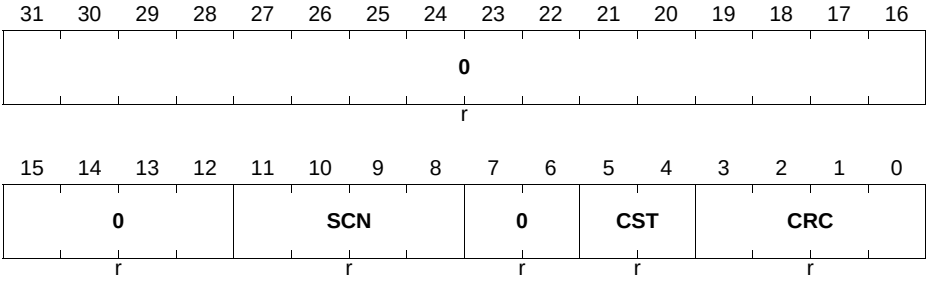


Field	Bits	Type	Description
WDLx	[15:0]	rw	**Watch Dog Timer Limit** for channel x .
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

Receiver Status Register

The Receive Status Register provides the status information of channel x.

RSRx (x = 0-3)
Receive Status Register x **($10C_H+40_H*x$)** **Reset Value: 0000 0000_H**

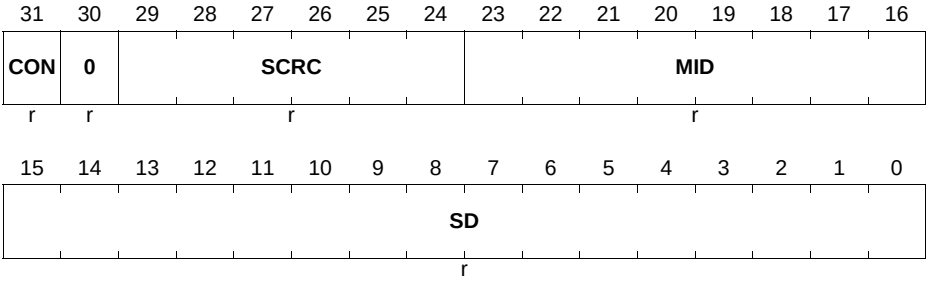


Field	Bits	Type	Description
CRC	[3:0]	r	**CRC** of last frame. CRC_0 is on bit position 0.
CST	[5:4]	r	**Channel Status** CST shows the current status of channel x. 00_B STOP Channel is disabled and can be configured 01_B INITIALIZED Channel is configured and enabled and no Synchronization / Calibration Pulse was received since last enable. 10_B RUNNING one or more Synchronization / Calibration Pulses were received and Frequency Range or Frequency Drift not or no longer in range. Fallback status from SYNCHRONIZED. 11_B SYNCHRONIZED Frequency Range and Frequency Drift in range
SCN	[11:8]	r	**Status and Communication Nibble** of last frame. SCN_0 is on bit position 8.
0	[7:6], [31:12]	r	**Reserved** Read as 0; should be written with 0.

Serial Data and Status Register

The Serial (Receive) Data and Status Register provides the data and status information of channel x.

SDSx (x = 0-3)
Serial Data and Status Register x (110_H+40_H*x) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
SD	[15:0]	r	**Serial Data** of last serial data frame. SD_0 is on bit position 0. If RCR.ESF is cleared 8 bits of data are available and bits [15:8] are zero. If RCR.ESF is set and if SDS.CON is cleared 12 bits of data are available and bits [15:12] are zero.
MID	[23:16]	r	**Message ID** of last serial data frame. ID_0 is on bit position 16. If RCR.ESF is cleared, or if SDS.CON is set, bits [23:20] are zero.
SCRC	[29:24]	r	**SCRC** CRC of last serial data frame. CRC_0 is on position 24. If RCR.ESF is cleared, bits [29:28] are always zero.
CON	31	r	**Configuration bit** of last serial frame. 0_B 12-bit data and 8-bit message ID 1_B 16-bit data and 4-bit message ID
0	30	r	**Reserved** Read as 0; should be written with 0.

22.2.4 Input and Output Control

Input and Output Control Register Functions

The Input and Output Control Register IOCRx determines for the SENT channel x:

for the receiver:

- the alternate input
- the filter depth
- the input signal polarity

for the SPC Unit

- the trigger source
- the output signal polarity

IOCRx (x = 0-3)
Input and Output Control Register x(114_H+40_H*x) **Reset Value: X000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXM	RXM	TRM	CTR	EC								ETS	ETS		
rh	rh	rh	rw	rh								rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFEG	CREG	FEG	REG	0	CEC	IIE	OIE	DEPTH				0		ALTI	
rw	rw	rh	rh	r	w	rw	rw	rw				r		rw	

Field	Bits	Type	Description
ALTI	[1:0]	rw	**Alternate Input Select** Selects the alternate input for channel y: 0000_B Alternate Input 0 selected 0001_B Alternate Input 1 selected $…_B$ … 0011_B Alternate Input 3 selected

Field	Bits	Type	Description
DEPTH	[7:4]	rw	**Digital Glitch Filter Depth** DEPTH determines the number of port input samples clocked with f_{pdiv} that are taken into account for the calculation of the floating average. The higher DEPTH is chosen to be, the longer the glitches that are suppressed and the longer the delay of the input signal introduced by this filter. 0000_B off, default 0001_B 1 T_{pdiv} 0010_B 2 0011_B 3 $\cdots_B$ … 1111_B 15
OIE	8	rw	**Output Inverter Enable Channel x** Selects the Pulse Polarity of the output of channel x 0_B Pulse polarity is active low 1_B Pulse polarity is active high
IIE	9	rw	**Input Inverter Enable Channel x** Selects the Pulse Polarity of the input of channel x 0_B Pulse polarity is active low 1_B Pulse polarity is active high
CEC	10	w	**Clear Edge Counter** If this bit is set, IOCR.EC is cleared. Always reads back as ‘0’.
REG	12	rh	**Rising Edge Glitch Flag for Channel x** Shows the status of the glitch detection of channel x 0_B No Glitch detected on rising edge 1_B Glitch detected on rising edge REG is cleared by setting CREG.
FEG	13	rh	**Falling Edge Glitch Flag for Channel x** Shows the status of the glitch detection of channel x 0_B No Glitch detected on falling edge 1_B Glitch detected on falling edge FEG is cleared by setting CFEG.
CREG	14	rw	**Clear Rising Edge Glitch Flag for Channel x** Clears the status flag REG 0_B REG is not cleared 1_B REG is cleared CREG always read zero.

Field	Bits	Type	Description
CFEG	15	rw	**Clear Falling Edge Glitch Flag for Channel x** Clears the status flag FEG 0_B FEG is not cleared 1_B FEG is cleared CFEG always read zero.
ETS	[19:16]	rw	**External Trigger Select** Selects the external trigger line if SCRx.TRIG is programmed to 11_B. 0000_BTRIG0 0001_BTRIG1 ... ... 3_D TRIG3 ... ... 1111_Breserved
EC	[27:20]	rh	**Edge Counter** This bit field contains a counter with saturation (stops at 0xFF). It is incremented with any falling edge that appears on the input pin selected by IOCR.ALTI. Note that this holds true in all states (STOP, INITIALIZED, RUNNING, SYNCHRONIZED). It is intended for debugging, in particular to find a bubbling idiot that sends before enabling the module.
CTR	28	rw	**Clear Trigger Monitor Flag for Channel x** Clears the status flag TRM 0_B TRM is not cleared 1_B TRM is cleared CTR always read zero. Reset value of CTR is 0.
TRM	29	rh	**Trigger Monitor Flag for Channel x** Shows the status of the trigger detection of channel x 0_B No Trigger detected 1_B Trigger detected (one or several) TRM is cleared by setting CTR. Reset value of TRM is 0.
RXM	30	rh	**Receive Monitor for Channel x** Shows the status of the receive signal of channel x after glitch filtering and inverted as specified by IIE. 0_B Current signal is low. 1_B Current signal is high. Reset value of RXM is X.

Field	Bits	Type	Description
TXM	31	rh	**Transmit Monitor for Channel x** Shows the status of the transmit signal of channel x inverted as specified by OIE. 0_B Current signal is low. 1_B Current signal is high. Reset value of TXM is X.
0	[3:2], 11	r	**Reserved** Read as 0; should be written with 0.

22.2.5 Receive Data Registers

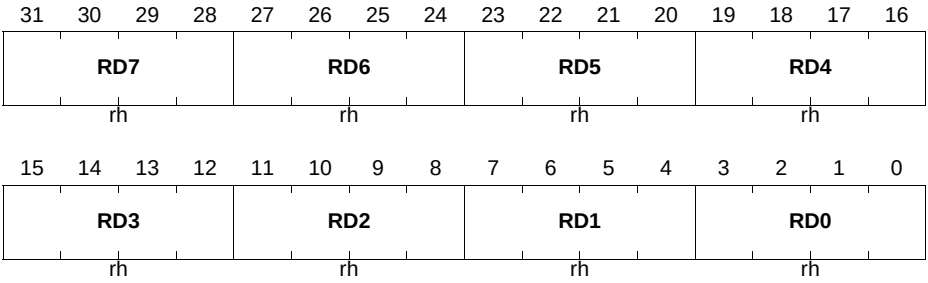
Receive Data Registers RDRx

The Receive Data Registers RDRx for channel x shows the data content of a received data frame. Register VIEWx is used to sort the nibbles.

Note: Register VIEW must be set up correctly to see all data nibbles of the frame! By default the application software set VIEW to 7654 3210$_H$.

Note: The internal receive buffer is always cleared (0x0000 0000$_H$) at each frame start. Thus unused nibbles are always read as zero.

RDRx (x = 0-3)
Receive Data Register x **(80_H+x*4_H)** **Reset Value: 0000 0000_H**

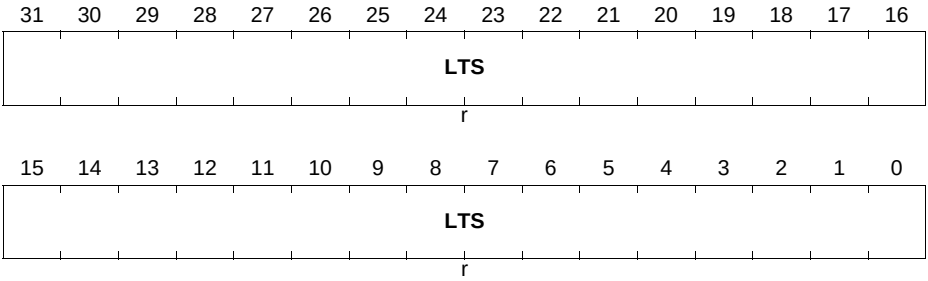


Field	Bits	Type	Description
RDy (y = 0-7)	[4*y+3 :4*y]	rh	**Receive Data Nibble y** RDy shows the nibble from the received frame that is sorted to this position. It can be selected by any of VIEWx.RDNPy (y = 0-7). By default all nibbles are sorted to RD0 as the reset value of VIEW is 0x0000 0000_H. I.e. at the end of frame reception RD0 contains the last data nibble of the frame.

Channel Receive Time Stamp Register

The SENT Channel x Receive Time Stamp Register contains read-only information about the time the last frame for channel x was received. Time is captured with the second falling edge, i.e. at the Status/Communication data pulse.

RTSx (x=0-3)
Receive Time Stamp Register x (A80$_H$+x*4$_H$) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
LTS	[31:0]	r	**Last Receive Time Stamp for Channel x** This bit field shows the time stamp of the last frame on channel x.

Receive Data View Register VIEWx

The Receive Data View Registers VIEWx stores the nibble pointers. They determine the sequence in which the received data nibbles are presented to the host. This reduces the SW effort to sort the nibbles.
The data nibble that is received first in the received frame is moved to the location given in register VIEW.RDNP0, the second to VIEW.RDNP1 and so on until VIEW.RDNP7.
If more than one VIEW.RDNPx point to a certain location in RDR, the last one will overwrite the previous ones.
Example: two 12 bit values are transmitted. One with highest significant nibble first and one with lowest significant nibble first. The frame looks like this: 456321$_H$. Note that the 1 is received as first data nibble and the 4 comes in as last data nibble.
The actual signal values are 0x123$_H$ and 0x456$_H$. By using VIEWx this can be sorted out into two 16bit values by HW. In the example VIEWx would be set to: 73 654 012$_H$. 73 is a dummy value and is not regarded if not more than 6 nibbles are received in a frame. The Register RDRx looks like this: 0x0456 0123$_H$ to the host.

In the example RDR nibbles 3 and 7 contain 0x0000$_B$ as the Receive Buffer is always cleared (0x0000 0000$_H$) before new data is received.

If a frame contains more than eight nibbles and the sorting can not be specified statically, VIEW can be set to e.g. 7654 3210_H.

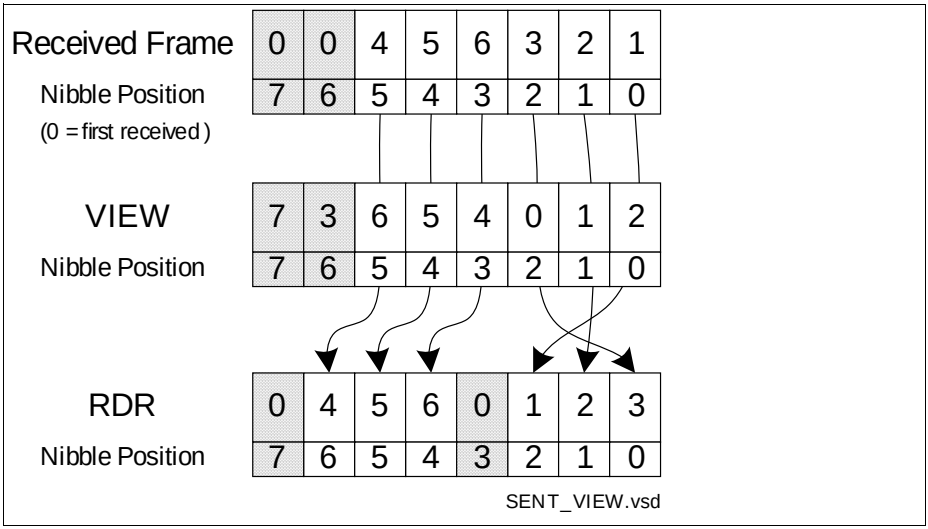


Figure 22-16 Functionality of VIEW Register

VIEWx (x = 0-3)

Receive Data View Register x ($11C_H$+40_H*x) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**RDNP7**			**0**	**RDNP6**			**0**	**RDNP5**			**0**	**RDNP4**		
r	rw			r	rw			r	rw			r	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**RDNP3**			**0**	**RDNP2**			**0**	**RDNP1**			**0**	**RDNP0**		
r	rw			r	rw			r	rw			r	rw		

Field	Bits	Type	Description
RDNPy (y = 0-7)	[4*y+2 :4*y]	rw	**Receive Data Target Nibble Pointer y** RDNPy points to the Nibble in Receive Data Register RDRx where the nibble y from the received frame is sorted to. Nibble 0 is the first data nibble in the frame. It gets moved to the position defined in RDNP0. And on. 000_B Nibble 0 selected 001_B Nibble 1 selected $\ldots_B$ … 111_B Nibble 7 selected *Note: RDNPy must be written before first frame reception. All RDNPy must have different values. (Higher RDNPy overwrite lower RDNPy.)*
0	3, 7, 11, 15, 19, 23, 27, 31	r	**Reserved** Read as 0; should be written with 0.

22.2.6 SPC Control

SPC Control Registers SCRx

The SPC Control Register SCR contains data to be transmitted during the sync pulse of the data frames. It contains as well the trigger control bits required for sending nibbles from the SENT module to the sensor / external SENT device.

The SPC Control Register is used to control the trigger mode and time base of the SPC channel transmission.

Data and the control bits are collected in this single register to ease transfer of multiple pulses in cases where dynamic switching of the trigger condition is required.

The SPC Control Register is build to control single pulse transfers only.

Thus it is possible to change the control settings of an individual channel from pulse to pulse as it is required in SPC mode “Bidirectional transmit”. Here it might be considered useful to change trigger mode between Mode 1 (immediately) and Mode 2 (falling edge of next Synchronization / Calibration Pulse with programmable delay).

In addition repeating transfers with the same control settings are supported. For a pulse transfer to be initiated a synchronization signal can be sufficient and no further SW intervention is required. Here the data can be changed (“ID-Selection” Mode) or simply left constant (“Sync” Mode). Only a HW trigger needs to be set up.

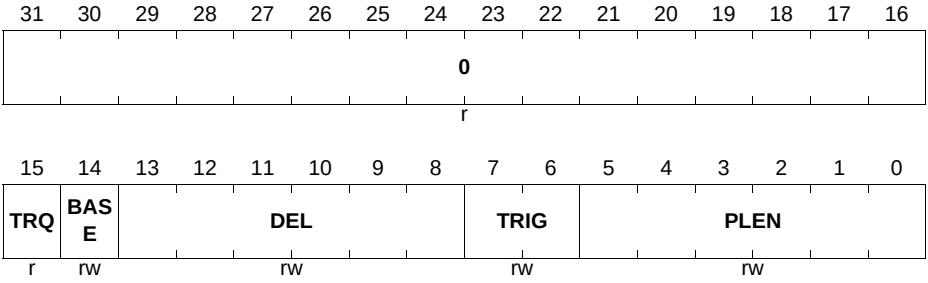
In Mode 0, SPC is deactivated for this channel. A write access to SCRx does not initiate an SPC pulse transmission. All state transmitter machines are initialized.

In Mode 1, an SPC pulse is sent, each time SPC Control Register SCRx is written to. If a transfer is ongoing, the channel waits automatically until the internal transmission register is ready. Transmit Buffer Underflow bit TBIx is set after data has been completely transferred (PLEN exceeded) and no new data was written to SCRx. After the data was transferred to the internal transmission register, interrupt TDIx signals that a new value can be written. INTSTATx.TDIx must be cleared by SW. Independently from this interrupt pending bit, a new interrupt pulse is generated on each transfer of an SPC pulse.
This mode is important for back to back transfers of several nibbles as in bidirectional SPC mode.

In Mode 2, an SPC pulse is sent, each time the first falling edge of any Synchronization / Calibration Pulse is received. In this mode, the programmable delay DEL is most useful. In SPC mode “Bidirectional Transmit” this mode is useful to synchronize the transmission. TDIx and TBIx work as in Mode 1.

In Mode 3, an SPC pulse is sent (with current DEL and PLEN) after each external trigger event (defined by IOCRx.ETS). This is most useful in SPC “sync” mode. TDIx and TBIx work as in Mode 1.

SCRx (x = 0-3)
SPC Control Register x **(118_H+40_H*x)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
PLEN	[5:0]	rw	**Pulse Length** Defines the length of the pulse in tick times. The time base is the measured tick time of the latest received frame if selected so by BASE. In case this measured tick time was invalid or not already available after enable of the channel, the nominal time base of the module is used. 000000_B Pulse length is 0 ticks 000001_B Pulse length is 1 tick ... ... 111111_B Pulse length is 63 ticks

Field	Bits	Type	Description
TRIG	[7:6]	rw	**Trigger Source and Mode Selection** Selects the Trigger Source and Mode. The internal sender state machine can be initialized by switching the channel off (TRIG is cleared) and on. This does not change the current register content. 00_B No Pulse is generated, OFF When cleared, an ongoing transfer is stopped immediately and the transmit output is driven recessive. 01_B Pulse starts immediately (no auto repetition) 10_B Pulse starts each time the first falling edge of any Synchronization / Calibration Pulse is received (auto repetition on next Sync. / Cal. Pulses) 11_B Pulse starts after each external trigger event. (auto repetition on next trigger) IOCRx.ETS selects the source of this event.
DEL	[13:8]	rw	**Delay Length** Selects how long the SPC pulse is delayed after the trigger condition. The time base is the measured tick time of the latest received frame if selected so by BASE. In case this measured tick time was invalid or not already available after enable of the channel, the nominal time base of the module is used. 000000_B Pulse is not delayed 000001_B Pulse is delayed by 1 tick ... ... 111111_B Pulse is delayed by 63 ticks
BASE	14	rw	**Time Base** Selects the Pulse Time Base 0_B Pulse is based on measured frequency of last Synchronization/Calibration Pulse 1_B Pulse is based on nominal frequency
TRQ	15	r	**Transfer Request in Progress** While an SPC Pulse is being sent this bit is set. Write access is ignored.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

22.2.7 Interrupt Control Registers

INTOV
Interrupt Overview Register **(14_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						IPC9	IPC8	IPC7	IPC6	IPC5	IPC4	IPC3	IPC2	IPC1	IPC0
r						rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
IPCy (y = 0-3)	y	rh	**Interrupt Pending on Channel y** If any interrupt requested flag is set for channel y in register INTSTATy AND the referring interrupt is enabled in INTENx then IPCy is set. It is automatically reset if all flags in INTSTATy are cleared for which the referring interrupt is enabled in INTENx. *Note: Not all IPC0-9 are available, the number of Interrupt Pending on Channel is equivalent to the SENT channels available, e.g 4 SENT channels has 4 Interrupt Pending IPC0-3 and vice versa.*
0	[31:4]	r	**Reserved** Read as 0.

Interrupt Status Register

The Interrupt Status Register INTSTATx contains status bits that show the status of any interrupt of SENT channel x.

Note: The bits are set independently from the referring Interrupt Enable in Register INTENx. Thus they can be used as status bits as well e.g. by a SW based on polling.

INTSTATx (x = 0-3)
Interrupt Status Register x **(120_H+40_H*x)** **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		WDI	SCRI	SDI	WSI	CRCI	NVI	NNI	FDI	FRI	TBI	TDI	RBI	RDI	RSI
r		rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
RSI	0	rh	**Receive Success Interrupt Request Flag** This bit is set at the successfully received end of a frame. Depending on bit RCRx.CDIS this indicates a successful check of the CRC. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time This bit can be cleared by bit INTCLRx.RSI. This bit can be set by bit INTSETx.RSI. This bit is set independently from INTENx.

Field	Bits	Type	Description
RDI	1	rh	**Receive Data Interrupt Request Flag** RDI is activated when a received frame is moved to a Receive Data Register RDR. Both RDI and RSI will be issued together in normal use cases where the frame size is not bigger than 8 nibbles and CRC is correct or not checked (if RCRx.CDIS is cleared). 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time This bit can be cleared by bit INTCLRx.RDI. This bit can be set by bit INTSETx.RDI. This bit is set independently from INTENx.
RBI	2	rh	**Receive Buffer Overflow Interrupt Request Flag** This bit is set after a frame has been received while the old one was not read from RDRx. I.e. the kernel wants to set any of the two interrupts RSI and RDI and finds any of these two interrupts already set.The old data is overwritten by the new data. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time This bit is NOT cleared by reading RDRx. This bit can be cleared by bit INTCLRx.RBI. This bit can be set by bit INTSETx.RBI. This bit is set independently from INTENx.
TDI	3	rh	**Transfer Data Interrupt Request Flag** This bit is set after the trigger condition was detected. Data to be transferred has been moved internally. Thus a new value can be written to SCRx. This can be used for back to back transfers. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time This bit is automatically cleared by writing SCRx. This bit can be cleared by bit INTCLRx.TDI. This bit can be set by bit INTSETx.TDI. This bit is set independently from INTENx.

Field	Bits	Type	Description
TBI	4	rh	**Transmit Buffer Underflow Interrupt Request Flag** This bit is set after data has been completely transferred (PLEN exceeded) and no new data was written to SCRx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time This bit is NOT cleared by writing SCRx. This bit can be cleared by bit INTCLRx.TBI. This bit can be set by bit INTSETx.TBI. This bit is set independently from INTENx.
FRI	5	rh	**Frequency Range Interrupt Request Flag** This bit is set after a Synchronization / Calibration pulse was received that deviates more than +- 25% from the nominal value. The referring data is ignored. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time This bit can be cleared by bit INTCLRx.FRI. This bit can be set by bit INTSETx.FRI. This bit is set independently from INTENx.
FDI	6	rh	**Frequency Drift Interrupt Request Flag** This bit is set after a subsequent Synchronization / Calibration pulse was received that deviates more than 1.5625% (1/64) from its predecessor. (See RCR.CFC) 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time This bit can be cleared by bit INTCLRx.FDI. This bit can be set by bit INTSETx.FDI. This bit is set independently from INTENx.

Field	Bits	Type	Description
NNI	7	rh	**Number of Nibbles Wrong Request Flag** This bit is set after a more nibbles have been received than expected or a Synchronization / Calibration Pulse is received too early thus too few nibbles have been received. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time This bit can be cleared by bit INTCLRx.NNI. This bit can be set by bit INTSETx.NNI. This bit is set independently from INTENx.
NVI	8	rh	**Nibbles Value out of Range Request Flag** This bit is set after a too long or too short nibble pulse has been received. I.e. value < 0 or value > 15. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time This bit can be cleared by bit INTCLRx.NVI. This bit can be set by bit INTSETx.NVI. This bit is set independently from INTENx.
CRCI	9	rh	**CRC Error Request Flag** This bit is set if the CRC fails. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time This bit can be cleared by bit INTCLRx.CRCI. This bit can be set by bit INTSETx.CRCI. This bit is set independently from INTENx.

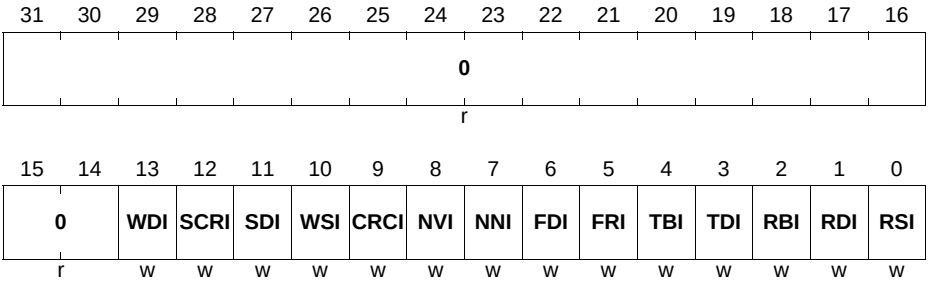
Field	Bits	Type	Description
WSI	10	rh	**Wrong Status and Communication Nibble Error Request Flag** In Short Serial Frame Mode (RCR.ESF is cleared), this bit is set if the Status and Communication nibble shows a start bit in a frame other than frame number n x 16. In Enhanced Serial Frame Mode this bit is without function. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time This bit can be cleared by bit INTCLRx.WSI. This bit can be set by bit INTSETx.WSI. This bit is set independently from INTENx.
SDI	11	rh	**Serial Data Receive Interrupt Request Flag** This bit is set after all serial data bits have been received via the Status and Communication nibble. Depending on bit RCRx.SCDIS this indicates a successful check of the CRC. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time This bit can be cleared by bit INTCLRx.SDI. This bit can be set by bit INTSETx.SDI. This bit is set independently from INTENx.
SCRI	12	rh	**Serial Data CRC Error Request Flag** This bit is set if the CRC of the serial message fails. In Enhanced Serial Message Format, this includes a check of the Serial Communication Nibble for correct 0 values of bit 3 in frames 7, 13 and 18. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time This bit can be cleared by bit INTCLRx.SCRI. This bit can be set by bit INTSETx.SCRI. This bit is set independently from INTENx.

Field	Bits	Type	Description
WDI	13	rh	**Watch Dog Error Request Flag** This bit is set if the Watch Dog Timer of the channel x expires. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time This bit can be cleared by bit INTCLRx.WDI. This bit can be set by bit INTSETx.WDI. This bit is set independently from INTENx.
0	[31:14]	r	**Reserved** Read as 0.

Interrupt Set Register

The Interrupt Set Register INTSETx contains control bits that trigger an interrupt pulse for any interrupt of SENT channel x.

INTSETx (x = 0-3)
Interrupt Set Register x **(124_H+40_H*x)** **Reset Value: 0000 0000_H**



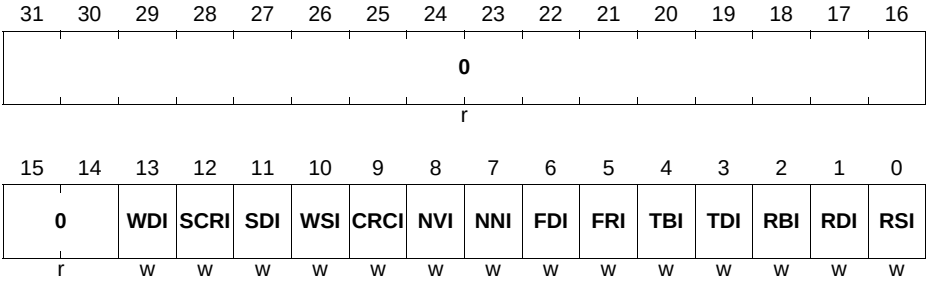
Field	Bits	Type	Description
RSI	0	w	**Set Interrupt Request Flag RSI** Setting this bit set bit INTSTATx.RSI. Clearing this bit has no effect. Reading this bit returns always zero.
RDI	1	w	**Set Interrupt Request Flag RDI** Setting this bit set bit INTSTATx.RDI. Clearing this bit has no effect. Reading this bit returns always zero.
RBI	2	w	**Set Interrupt Request Flag RBI** Setting this bit set bit INTSTATx.RBI. Clearing this bit has no effect. Reading this bit returns always zero.
TDI	3	w	**Set Interrupt Request Flag TDI** Setting this bit set bit INTSTATx.TDI. Clearing this bit has no effect. Reading this bit returns always zero.
TBI	4	w	**Set Interrupt Request Flag TBI** Setting this bit set bit INTSTATx.TBI. Clearing this bit has no effect. Reading this bit returns always zero.

Field	Bits	Type	Description
FRI	5	w	**Set Interrupt Request Flag FRI** Setting this bit set bit INTSTATx.FRI. Clearing this bit has no effect. Reading this bit returns always zero.
FDI	6	w	**Set Interrupt Request Flag FDI** Setting this bit set bit INTSTATx.FDI. Clearing this bit has no effect. Reading this bit returns always zero.
NNI	7	w	**Set Interrupt Request Flag NNI** Setting this bit set bit INTSTATx.NNI. Clearing this bit has no effect. Reading this bit returns always zero.
NVI	8	w	**Set Interrupt Request Flag NVI** Setting this bit set bit INTSTATx.NVI. Clearing this bit has no effect. Reading this bit returns always zero.
CRCI	9	w	**Set Interrupt Request Flag CRCI** Setting this bit set bit INTSTATx.CRCI. Clearing this bit has no effect. Reading this bit returns always zero.
WSI	10	w	**Set Interrupt Request Flag WSI** Setting this bit set bit INTSTATx.WSI. Clearing this bit has no effect. Reading this bit returns always zero.
SDI	11	w	**Set Interrupt Request Flag SDI** Setting this bit set bit INTSTATx.SDI. Clearing this bit has no effect. Reading this bit returns always zero.
SCRI	12	w	**Set Interrupt Request Flag SCRI** Setting this bit set bit INTSTATx.SCRI. Clearing this bit has no effect. Reading this bit returns always zero.
WDI	13	w	**Set Interrupt Request Flag WDI** Setting this bit set bit INTSTATx.WDI. Clearing this bit has no effect. Reading this bit returns always zero.
0	[31:14]	r	**Reserved** Read as 0; should be written with 0.

Interrupt Clear Register

The Interrupt Clear Register INTCLRx contains control bits that clear the status of any interrupt of SENT channel x.

INTCLRx (x = 0-3)
Interrupt Clear Register x **(128_H+40_H*x)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
RSI	0	w	**Clear Interrupt Request Flag RSI** Setting this bit clears bit INTSTATx.RSI. Clearing this bit has no effect. Reading this bit returns always zero.
RDI	1	w	**Clear Interrupt Request Flag RDI** Setting this bit clears bit INTSTATx.RDI. Clearing this bit has no effect. Reading this bit returns always zero.
RBI	2	w	**Clear Interrupt Request Flag RBI** Setting this bit clears bit INTSTATx.RBI. Clearing this bit has no effect. Reading this bit returns always zero.
TDI	3	w	**Clear Interrupt Request Flag TDI** Setting this bit clears bit INTSTATx.TDI. Clearing this bit has no effect. Reading this bit returns always zero.
TBI	4	w	**Clear Interrupt Request Flag TBI** Setting this bit clears bit INTSTATx.TBI. Clearing this bit has no effect. Reading this bit returns always zero.

Field	Bits	Type	Description
FRI	5	w	**Clear Interrupt Request Flag FRI** Setting this bit clears bit INTSTATx.FRI. Clearing this bit has no effect. Reading this bit returns always zero.
FDI	6	w	**Clear Interrupt Request Flag FDI** Setting this bit clears bit INTSTATx.FDI. Clearing this bit has no effect. Reading this bit returns always zero.
NNI	7	w	**Clear Interrupt Request Flag NNI** Setting this bit clears bit INTSTATx.NNI. Clearing this bit has no effect. Reading this bit returns always zero.
NVI	8	w	**Clear Interrupt Request Flag NVI** Setting this bit clears bit INTSTATx.NVI. Clearing this bit has no effect. Reading this bit returns always zero.
CRCI	9	w	**Clear Interrupt Request Flag CRCI** Setting this bit clears bit INTSTATx.CRCI. Clearing this bit has no effect. Reading this bit returns always zero.
WSI	10	w	**Clear Interrupt Request Flag WSI** Setting this bit clears bit INTSTATx.WSI. Clearing this bit has no effect. Reading this bit returns always zero.
SDI	11	w	**Clear Interrupt Request Flag SDI** Setting this bit clears bit INTSTATx.SDI. Clearing this bit has no effect. Reading this bit returns always zero.
SCRI	12	w	**Clear Interrupt Request Flag SCRI** Setting this bit clears bit INTSTATx.SCRI. Clearing this bit has no effect. Reading this bit returns always zero.
WDI	13	w	**Clear Interrupt Request Flag WDI** Setting this bit clears bit INTSTATx.WDI. Clearing this bit has no effect. Reading this bit returns always zero.
0	[31:14]	r	**Reserved** Read as 0; should be written with 0.

Interrupt Enable Register

The Interrupt Enable Register INTENx contains control bits that enable the interrupt source of any interrupt of SENT channel x.

Note: The Interrupt Status bits in register INTSTATx are set independently from the Interrupt Enable in Register INTENx.

INTENx (x = 0-3)
Interrupt Enable Register x **($12C_H$+40_H*x)** **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		WDI	SCRI	SDI	WSI	CRCI	NVI	NNI	FDI	FRI	TBI	TDI	RBI	RDI	RSI
r		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
RSI	0	rw	**Enable Interrupt Request RSI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
RDI	1	rw	**Enable Interrupt Request RDI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
RBI	2	rw	**Enable Interrupt Request RBI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source

Field	Bits	Type	Description
TDI	3	rw	**Enable Interrupt Request TDI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
TBI	4	rw	**Enable Interrupt Request TBI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
FRI	5	rw	**Enable Interrupt Request FRI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
FDI	6	rw	**Enable Interrupt Request FDI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
NNI	7	rw	**Enable Interrupt Request NNI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
NVI	8	rw	**Enable Interrupt Request NVI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
CRCI	9	rw	**Enable Interrupt Request CRCI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source

Field	Bits	Type	Description
WSI	10	rw	**Enable Interrupt Request WSI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
SDI	11	rw	**Enable Interrupt Request SDI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
SCRI	12	rw	**Enable Interrupt Request SCRI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
WDI	13	rw	**Enable Interrupt Request WDI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
0	[31:14]	r	**Reserved** Read as 0; should be written with 0.

Interrupt Node Pointer Register

The Interrupt Node Pointer Register INPx contains the node pointers of SENT channel x.

Note: Node Pointer ERRI is one single node pointer for the following error interrupts:

- FRI
- FDI
- NNI
- NVI
- CRCI
- WSI
- SCRI

INPx (x = 0-3)

Interrupt Node Pointer Register x (130_H+40_H*x) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WDI				SDI				ERRI				TBI			
rw				rw				rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDI				RBI				RDI				RSI			
rw				rw				rw				rw			

Field	Bits	Type	Description
RSI	[3:0]	rw	**Interrupt Node Pointer for Interrupt RSI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.RSI (if enabled by bit INTENx.RSI). 0000_B Interrupt node 0 is selected 0001_B Interrupt node 1 is selected 0010_B Interrupt node 2 is selected 0011_B Interrupt node 3 is selected 0100_B Trigger Output TRIGO 0 is selected 0101_B Trigger Output TRIGO 1 is selected 0110_B Reserved, do not use ... ... 1111_B Reserved, do not use

Field	Bits	Type	Description
RDI	[7:4]	rw	**Interrupt Node Pointer for Interrupt RDI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.RDI (if enabled by bit INTENx.RDI). For bit field definition, see RSI.
RBI	[11:8]	rw	**Interrupt Node Pointer for Interrupt RBI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.RBI (if enabled by bit INTENx.RBI). For bit field definition, see RSI.
TDI	[15:12]	rw	**Interrupt Node Pointer for Interrupt TDI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.TDI (if enabled by bit INTENx.TDI). For bit field definition, see RSI.
TBI	[19:16]	rw	**Interrupt Node Pointer for Interrupt TBI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.TBI (if enabled by bit INTENx.TBI). For bit field definition, see RSI.
ERRI	[23:20]	rw	**Interrupt Node Pointer for Interrupt FRI, FDI, NNI, NVI, CRCI, WSI, SCRI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.FRI (if enabled by bit INTENx.FRI) or INTSTATx.FDI (if enabled by bit INTENx.FDI) or INTSTATx.NNI (if enabled by bit INTENx.NNI) or INTSTATx.NVI (if enabled by bit INTENx.NVI) or INTSTATx.CRCI (if enabled by bit INTENx.CRCI) or INTSTATx.WSI (if enabled by bit INTENx.WSI) or INTSTATx.SCRI (if enabled by bit INTENx.SCRI) For bit field definition, see RSI.
SDI	[27:24]	rw	**Interrupt Node Pointer for Interrupt SDI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.SDI (if enabled by bit INTENx.SDI). For bit field definition, see RSI.

Field	Bits	Type	Description
WDI	[31:28]	rw	**Interrupt Node Pointer for Interrupt WDI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.WDI (if enabled by bit INTENx.WDI). For bit field definition, see RSI.

22.3 SENT Module Implementation

This section describes the SENT module interface as it is implemented in the TC21x/TC22x/TC23x. It especially covers clock control, port and on-chip connections, interrupt control, and address decoding.

22.3.1 Interface Connections of the SENT Module

Figure 22-17 shows the TC21x/TC22x/TC23x-specific implementation details and interconnections of the SENT module.

The SENT module is supplied with a separate clock control, address decoding, and interrupt control logic. Outputs of the GTM module are connected to the 4 timer inputs.

The serial data inputs of the receive channels of the SENT module as well as the SPC outputs (SPCn) are connected to GPIO lines. If SPC outputs are used, they are usually mapped to the same port pin like the referring SENT data input line as this minimizes the pin count requirement.

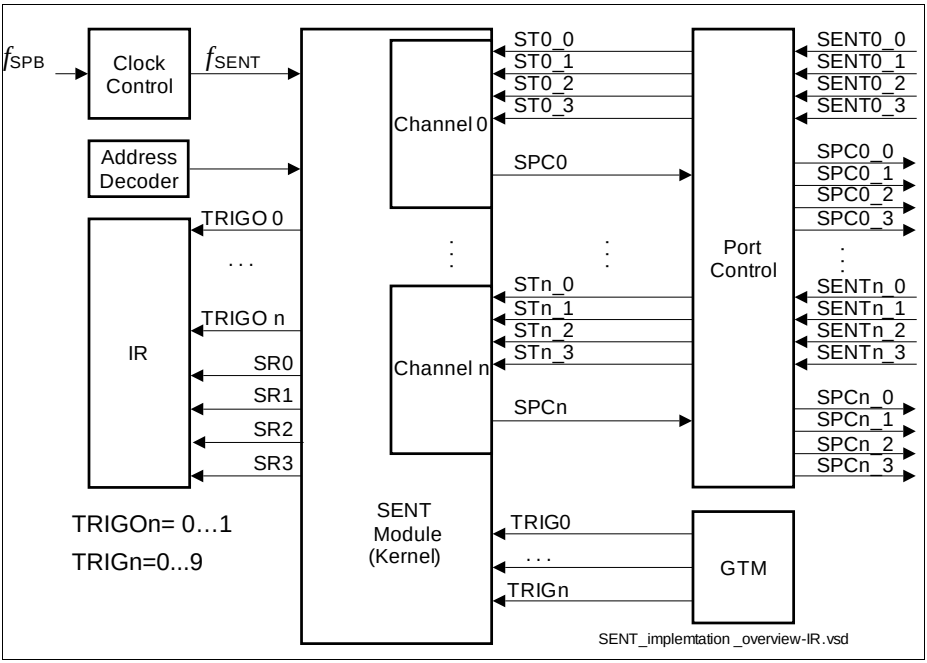


Figure 22-17 SENT Module Implementation and Interconnections

22.3.1.1 On-Chip Connections

This section describes the on-chip connections of the SENT module.

22.3.1.2 Interrupt and DMA Controller Service Requests

The trigger outputs of the SENT module are connected via the Interrupt router. The request lines are connected as shown in **Table 22-9**.

Table 22-9 Service Request Lines of SENT

INP value	Request Line	Connected to	Description
0000_b	SR0	SRC_SENT0	Interrupt Router SENT Request 0
0001_b	SR1	SRC_SENT1	Interrupt Router SENT Request 1
0010_b	SR2	SRC_SENT2	Interrupt Router SENT Request 2
0011_b	SR3	SRC_SENT3	Interrupt Router SENT Request 3
0100_b	TRIGO0	SRC_SENT4	Not Connected
0101_b	TRIGO1	SRC_SENT5	Not Connected
0110_b	TRIGO2	SRC_SENT6	Not Connected
0111_b	TRIGO3	SRC_SENT7	Not Connected
1000_b	TRIGO4	SRC_SENT8	Not Connected
1001_b	TRIGO5	SRC_SENT9	Not Connected
1010_b	TRIGO6	SRC_SENT10	Not Connected
...	...	...	...
1111_b	TRIGO11	SRC_SENT15	Not Connected

Trigger Inputs

The module has 4 Sent Channels and the same number of trigger inputs which can be randomly chosen by programming IOCRx.ETS. The trigger inputs (TRIG[3:0]) of the SENT module are connected to the GTM as shown in **Table 22-10**. n = [0 .. 3]

Table 22-10 Trigger Input Lines of SENT

Request Line	Connected to	Description
TRIG0	TRIG0	GTM.ADC_0_TRIG_0
TRIG1	TRIG1	GTM.ADC_1_TRIG_0
...	...	...

Table 22-10 Trigger Input Lines of SENT

Request Line	Connected to	Description
TRIG2	TRIG2	GTM.ADC_2_TRIG_0[1)]
TRIG3	TRIG3	GTM.ADC_3_TRIG_0[1)]

1) These connections are not available on TC23x PD and are available on TC23x ED only, as TC23x PD has only ADC 0/1 while TC23x ED has ADC0/1/2/3.

22.3.2 SENT Module-Related External Registers

The registers listed in **Figure 22-18** are external of the SENT module kernel but must be programmed for proper operation of the SENT module.

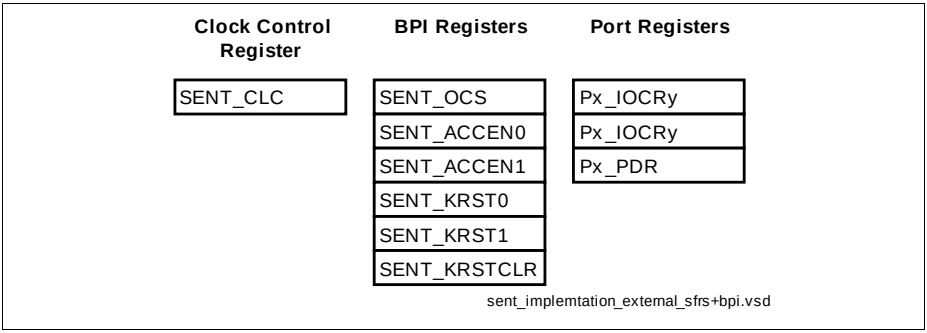


Figure 22-18 SENT Implementation-specific Special Function Registers

22.3.2.1 Port Control

The SENT input channels are overlaid with standard ADC channels as they are replacing former analog signals. Each channel is connected to two ADC channels that can be chosen alternatively. In addition each channel is connected to general purpose I//O lines as well. Each channel is connected to two different general purpose I/O lines that can be chosen alternatively. The selection from on of these 4 alternatives is done by application SW by programming SENT_IOCRx.ALTI respectively.

The SPC output lines are connected to the same I/O ports as the referring input channel. The ADC channels do not provide output functionality. The SPC I/O ports are controlled in the port logic (see also **Figure 22-17**). The following port control operations and selections must be executed for these I/O lines:

- Input/output function selection (Port IOCR registers)
- Pad driver characteristics selection for the outputs (Port PDR registers)

Input/Output Function Selection

SENT can be used in three different ways:

- ADC input (dedicated for ADC) overlaid with SENT digital input and no output option on this pin.
- SENT configured digital I/O lines (open drain, no push/pull, uses SDIR, single pin hardware direction control HW_DIR, SDIR is controlled by the data value: 0 = active out, 1 = passive/open drain, input)

- Standard general purpose input/output function on two lines for each channel (input on ADC alternative 1 or 2, or on I/O port alternative 1 while the output is chosen to be on I/O ports alternative 2.

The SENT module overlays dedicated analog to digital converter (ADC) pins as digital input port.

SENT physical layer uses 5V signal voltage levels. All SENT inputs are mapped to 5V compatible ADC pads. Vddm must be supplied with 5V for SENT operation.

The SENT module can be configured to use input/output ports configured for use with SENT. These control settings for the port pins differ from the standard general purpose I/O lines in so far as

- they are controlled by the SENT module output data via their HW_DIR line
- they are configured to use no push/pull devices and to work in open drain mode if the output function is selected by the HW_DIR line of the port

The SENT module can be configured to use standard (push pull) general purpose I/O lines as well. Different port pins can be selected for input and for output. This allows the use of external transceiver devices.

The port input/output control registers contain the bit fields that select the digital output and input driver characteristics such as port direction (input/output) with alternate output selection, pull-up/down devices, and open-drain selections. The I/O lines for the SENT module are controlled by the Port input/output control registers shown below.

Table 22-11 shows an overview how bits and bit fields must be programmed for the required I/O functionality of the SENT I/O lines.

Table 22-11 SENT I/O Control Selection and Setup

SENT Channel	Port Lines	Input Select Register	Input/Output Control Register Bits	I/O
0	SENT0A / P40.11	SENT_IOCR0.ALTI = 0000_B	P40_PDISC.PDIS11 = 0_B	I
	SENT0B / P00.1	SENT_IOCR0.ALTI = 0001_B	P00_IOCR0.PC1 = $0XXXX_B$	I
	SENT0C / P02.8	SENT_IOCR0.ALTI = 0010_B	P02_IOCR8.PC8 = $0XXXX_B$	I
		SENT_IOCR0.ALTI = 0001_B		O

Table 22-11 SENT I/O Control Selection and Setup (cont'd)

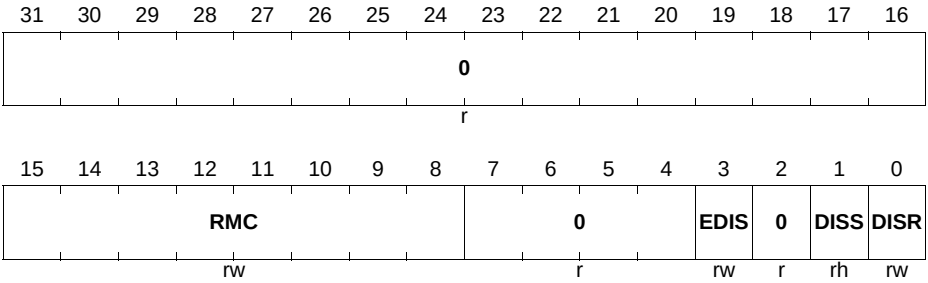
SENT Channel	Port Lines	Input Select Register	Input/Output Control Register Bits	I/O
1	SENT1A / P41.0	SENT_IOCR1.ALTI = 0000_B	P41_PDISC.PDIS0 = 0_B	I
	SENT1B / P00.2	SENT_IOCR1.ALTI = 0001_B	P00_IOCR0.PC2 = 0XXXX$_B$	I
	SENT1C / P02.7	SENT_IOCR1.ALTI = 0010_B	P02_IOCR4.PC7 = 0XXXX$_B$	I
		SENT_IOCR1.ALTI = 0010_B		O
2	SENT2A / P41.2	SENT_IOCR2.ALTI = 0000_B	P41_PDISC.PDIS2 = 0_B	I
	SENT2B / P00.3	SENT_IOCR2.ALTI = 0001_B	P00_IOCR0.PC3 = 0XXXX$_B$	I
	SENT2C / P02.6	SENT_IOCR2.ALTI = 0010_B	P02_IOCR4.PC6 = 0XXXX$_B$	I
		SENT_IOCR2.ALTI = 0001_B		O
3	SENT3A / P41.3	SENT_IOCR3.ALTI = 0000_B	P41_PDISC.PDIS3 = 0_B	I
	SENT3B / P00.4	SENT_IOCR3.ALTI = 0001_B	P00_IOCR4.PC4 = 0XXXX$_B$	I
	SENT3C / P02.5	SENT_IOCR3.ALTI = 0010_B	P02_IOCR4.PC5 = 0XXXX$_B$	I
		SENT_IOCR3.ALTI = 0001_B		O

22.3.3 BPI_FPI Module Registers

Clock Control Register (CLC)

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI_FPI for the module. Where a module kernel is connected to the CLC clock control interface, CLC controls the module clock, sleep mode and disable mode for the module.

SENT_CLC
Clock Control Register (00_H) Reset Value: $0000\ 0003_H$



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module.
EDIS	3	rw	**Sleep Mode Enable Control** Used to control module's sleep mode.
RMC	[15:8]	rw	**8-bit Clock Divider Value in RUN Mode**
0	[31:16], [7:4], 2	r	**Reserved** Read as 0; should be written with 0.

Note: The number of module clock cycles (wait states) which are required by the kernel to execute a read or write access depends on the selected CLC clock frequency, which is selected via bit field RMC in the CLC register. Therefore, increasing CLC.RMC may result in a longer FPI Bus read cycle access time for kernel registers and can also slow down the write throughput to the kernel registers.

13. *After a hardware reset operation, the f_{SENT} and $f_{fracdiv}$ clocks are switched off and the SENT module is disabled (DISS set).*

OCDS Control and Status Register (OCS)

The OCDS Control and Status (OCS) register is cleared by Debug Reset. The OCS register can only be written when the OCDS is enabled. If OCDS is being disabled, the OCS register value will not change. When OCDS is disabled the OCS suspend control is ineffective. Write access is 32bit wide only and requires Supervisor Mode.

OCS
OCDS Control and Status (E8$_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		SUS STA	SUS _P	SUS				0							
r		rh	w	rw				r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
SUS	[27:24]	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0_H Will not suspend 1_H Hard suspend. Clock is switched off immediately. (SPC low pulse breaks immediately, Port Pin keeps the last value if suspend state is entered) 2_H Soft suspend (SPC low pulse will be finished before suspend state is entered) **others**, reserved
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	[23:0], [31:30]	r	**Reserved** Read as 0; must be written with 0.

Access Enable Register (ACCEN0)

The Access Enable Register 0 controls write access[1)] for transactions with the on chip bus master TAG ID 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000_B, EN1 -> TAG ID 000001_B, ... , EN31 -> TAG ID 011111_B.

ACCEN0
Access Enable Register 0 (FC_H) Reset Value: FFFF FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	EN30	EN29	EN28	EN27	EN26	EN25	EN24	EN23	EN22	EN21	EN20	EN19	EN18	EN17	EN16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n = 0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

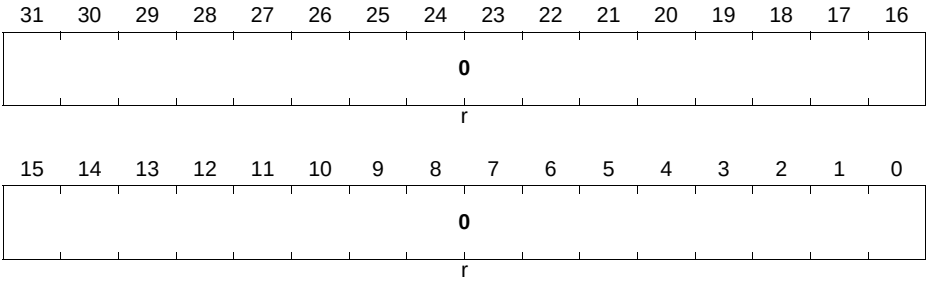
Access Enable Register (ACCEN1)

The Access Enable Register 1 controls write access[1)] for transactions with the on chip bus master TAG ID 100000_B to 111111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the AURIX devices.

1) The BPI_FPI Access Enable functionality controls only write transactions to the CLC, OCS, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000_B, EN1 -> TAG ID 100001_B, ... , EN31 -> TAG ID 111111_B.

ACCEN1
Access Enable Register 1 **($F8_H$)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
0	[31:0]	r	**Reserved** Read as 0; should be written with 0.

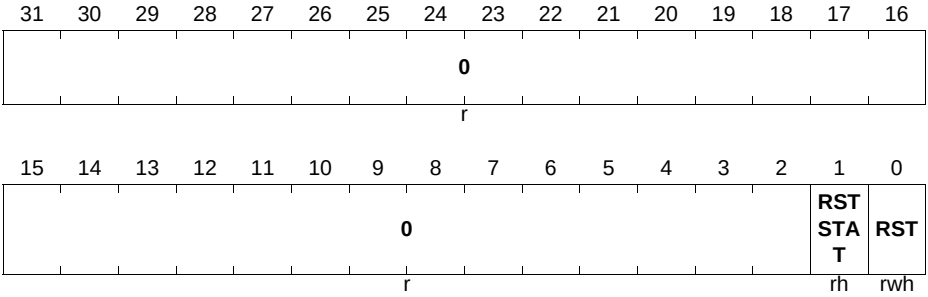
Kernel Reset Register 0 (KRST0)

The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers.The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the KRSTCLE.CLR register bit.

Note: During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

KRST0
Kernel Reset Register 0 **($F4_H$)** **Reset Value: 0000 0000$_H$**

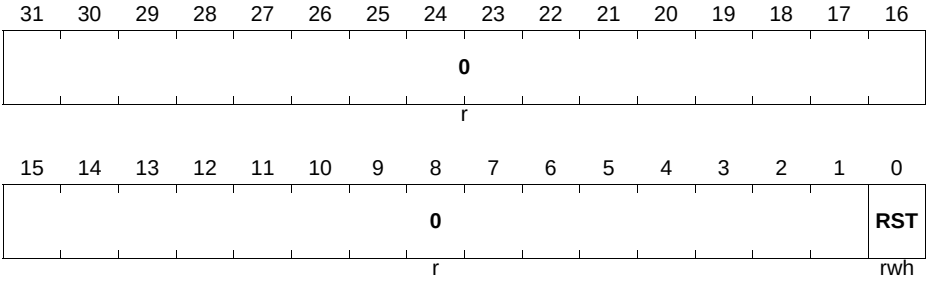


Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed.
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. 0_B No kernel reset was executed 1_B Kernel reset was executed This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register.
0	[31:2]	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1 (KRST1)

The Kernel Reset Register 1 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset registers. The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

KRST1
Kernel Reset Register 1 **($F0_H$)** **Reset Value: 0000 0000$_H$**



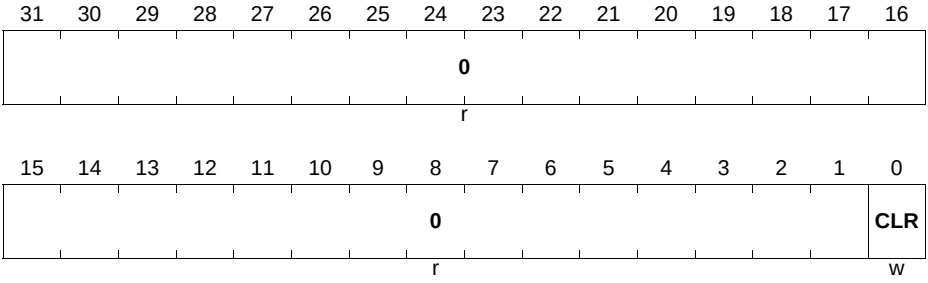
Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Status Clear Register (KRSTCLR)

The Kernel Reset Status Clear register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR

Kernel Reset Status Clear Register (EC_H) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT Read always as 0.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

22.4 Revision History

This User's Manual is based on: SAE Standard "SENT **Revision J2716 JAN2010**".

Table 22-12 Revision History

Version Number	Changes to Previous Version
Rev_0.1D	First Draft (Bullet Points ++)
Rev_0.2D	Second Draft (Bullet Points ++)
Rev_0.3D	First Review Version
Rev_0.4D	• RCRx.NINT removed as unnecessary FIFO support • Spelling Checked • Renamed INTNP to INP (request DE) • One common error node pointer in INP for FRI, FDI, NNI, NVI, CRCI, WSI, SCRI (reduce DE effort) • Interrupt Node Pointer in INPx increased from 2 bit to 4 bit • Renamed PVAL to PDIV in register CPDRx • RCRx.ASP "Additional Sync/Cal Pulse expected" deleted, not required • RCRx.ACE "Alternate CRC Mode" added, request customer • cleanup Register SCR • ISR0 (Input select register) deleted • AltIn and Filter DEPTH moved to IOCRx • SCRx.DEL moved to SDRx • SCRx deleted, BASE and TRQ moved to SDR • Restructured Chapters (interrupts after functional registers, implementation chapter cleaned up) • Simplified Baud rate generation (Fdtick removed)
Rev_0.5D	• Clean up Glitch filter Chapter • Detailed Baud Rate Generation • Added Address overview table

Table 22-12 Revision History (cont'd)

Version Number	Changes to Previous Version
Rev_0.6D	• Channel **Fractional Divider with variable divider** instead of variable dividend • Glitch filter based on F_pdiv and no longer on F_sent • Glitch detection added • Pre divider = PDIV and no longer PDIV + 1 • Compressed address overview table • Interrupt generation updated with figure • Trigger outputs renamed (RSI and SDI, no more TRIGO) and completed • RDI simplified - now independent from RSI • ETS wording corrected • INTOV now only updated for enabled interrupts (before: any change) • Renamed "SPC Data Register SDR" to "SPC Control Register SCR", it contains both data and control information • TDI wording optimized • External Registers Overview corrected • KSCFG removed
Rev_1.0D	First complete revision (To be done)
Rev_1.1D	• Moved Base Address to F032_1000 - F032_19FF (10x256 bytes) • Serial message ID added. SDSx.SCN moved to [19:16], SDSx.SCRC moved to [15:12], SDSx.MID placed at [11:8]. • RSI and TDI as trigger outputs replaced by 8 Trigger Outputs programmable in the INPs. • implemented first port mapping proposal from product
Rev_1.2	• Added hint to RCR.CEN that FSMs are initialized during enable • Port mapping updated to latest changes of CW08/47

Table 22-12 Revision History (cont'd)

Version Number	Changes to Previous Version
Rev_1.3	• Fixed INTENx Bit descriptions: all bits are rw and high active (typo) • added to feature list: "Programmable Nibble sorting to support LSN or HSN first and relief CPU" (was missing) • SENT_FDR.DM[1:0] added description of all 4 states of this bit field. (same as system level) • IOCR.DEPTH corrected f_{SENT} sent to f_{pdiv} (longer glitch tolerance) • Add Freeze-Disable Bit [10] to SENT_FDR (system requirement) • Pre divider factor is (PDIV + 1) and no longer PDIV (ease design) • Resulting Channel Fractional Divider is (DIV + 1) and no longer DIV • Add IOCR.TXM, RXM and TRM monitor bits (ease of validation) • RDRx added explanation: unused nibbles are shown as '0' (for clarity) • RDI and RSI: Read clears these bits NOT (request from AE) • Renamed Transmit Buffer Overflow to UNDER-Flow with change to respective functionality (data completely transmitted without new write) • RCRx.CFC: Wording improved and details added
Rev_1.4	• Added column Access Modes to Register Table • Updated **Figure 22-2**, **Figure 22-3**, **Figure 22-4** (beautification) • The clock signal f_{pdiv} of a channel must always be at least 20 (old value 10) times the nominal tick frequency, to cater for worst case. • INPx: Added reserved values • Corrected interrupt structure figure, no more separate DMA TRIGO lines. • RCRx, VIEWx and CFDRx Reset value corrected to 0x0000 0000 hex • Added explanations to RDRx and VIEWx • AI00050230 - SENT_FDR register located at "unusual" address: Moved CLC to offset 0x0Ch for consistency with other modules • Reduced ETS to 3 bit and ALTI to 2 bit width • "Zero Nibble Insertion in CRC" covered. See RCR.CRZ. • UTP AI00050205 - "Extended Serial Frame" covered. Added description of extended serial frame. SDS.SCN moved to RSR.SCN (Allows for an aligned representation of ID and Data as well as a fixed position for SCN) SDS.SD, SDS.MID, SDS.SCRC enlarged (to 16, 8 and 6 bit width) SDS.CON and RCR.ESF added • Resulting Channel Fractional Divider is DIV and no longer (DIV + 1)

Table 22-12 Revision History (cont'd)

Version Number	Changes to Previous Version
Rev_1.5	• Corrected TBIx description on Page 50. • SCR.TRQ corrected (deleted "Reads back zero") • Added hint to definition of Alternate CRC Mode (ACE) • Detailed RBI behavior. It is set if kernel wants to set RSI or RDI and finds RSI or RDI already set. • Detailed that RCR.CEN resets the receiver state machines only while SCR.TRIG resets the sender state machines. • RCR.WSI updated wrt. the special case "extended serial frame" (2009-06-24) • INSTAT.SCRI detailed wrt.: CRC check must include check for correct 0 values in Extended Serial Frame Format (2009-06-24) • Message tick time prolonged to 90 µs according to new standard (2009-06-24) • RCR.CFC (successive calibration pulse detection) adopted to new standard. (2009-06-24) • Updated INTSTAT.FDI wrt.: new additional error diagnostics (total frame length variation and ration calibration pulse/total frame length) in new standard. (2009-06-24)
Rev_1.6	• Added bit RCR.IDE (Ignore Drift Error) to cater for rare triggers by SPC. • Removed SV protection from SENT_FDR. • VIEW.RDNP must be set before reception. • SCR.TRQ spec' ed not active from request to pulse start. • Added "alternative CRC" spec from TLE4998 . • Changed wording for RBI. • New additional error diagnostics (total frame length variation and ration calibration pulse/total frame length) removed.
Rev_1.7	• Added note: RCR.CEN must be cleared before changing CPDR.PDIV or CFDR.DIV. • SCRI typo: frame 8, not 7 contains a zero value. • **Chapter 22.1.5** typo corrected: replaced f_{SENT} with $f_{fracdiv}$ • **Chapter 22.1.3.1** note added on *RCR.CRZ.* • Added hint to Chapter "Module Clock Generation" for details on CLC. • Updated Port connections and Trigger Output connections
Rev_1.8	• Changed functionality of VIEW. • Changed functionality of WSI.
Rev_1.9	
Rev_1.10	

Table 22-12 Revision History (cont'd)

Version Number	Changes to Previous Version
Rev_1.11	
Rev_1.12	•
Rev_1.13	
Rev_1.14	
Rev_1.15	•
Rev_1.16	•
Rev_1.17	•
Rev_1.18	• Modified **IOCRx (x = 0-3)** reset value to X000 0000 with additional text on IOCRx.CTR, IOCRx.TRM reset value is 0 and IOCRx.TXM, IOCRx.RXM reset value is X. •
Rev_1.19	• Removed on Figure 1-1, Figure 1-26 obsolete signal HW_DIR. • Added List of Access Protection Abbreviations on **Table 22-8** • Modified on **Table 22-8**, SENT_INTSTAT, SENT_RTS, SENT_INTOV, SENT_RSR, write access to BE (Bus Error). •
Rev_1.20	•
Rev_1.21	•
Rev_1.22	
Rev_1.23	•
Rev_1.24	• .
Rev_1.25	•
Rev_1.26	•
Rev_1.27	• •
Rev_1.28	•
Rev_1.29	•
Rev_1.30	•
Rev_1.31	•
Rev_1.32	•

Table 22-12 Revision History (cont'd)

Version Number	Changes to Previous Version
Rev_1.33	• Modified for below: – Added Note on INTOV, Interrupt Overview Register "Not all IPC0-9 are available, the number of Interrupt Pending on Channel is equivalent to the SENT channels available, e.g 4 SENT channels has 4 Interrupt Pending IPC0-3 and vice versa." – Replaced "Fig 1-7 Standard Serial Data Encoding" in .emf with same "Table 1-1 Standard Serial Data Encoding" as framemaker table. – Replaced "Fig 1-8 Serial Data Frame" in .emf with same "Table 1-2 "Serial Data Frame" as framemaker table. – Replaced "Fig 1-9 Extended Serial Data Encoding" in .emf with same "Table 1-3 Extended Serial Data Encoding" as framemaker table. – Replaced "Fig1-11 Extended Serial Data Frame" in .emf with same "Table 1-4 Extended Serial Data Frame" – Replaced "Fig 1-12 Configuration Bit = 0" in .emf with same "Table 1-5 Configuration Bit = 0" – Replaced "Fig 1-13 Configuration Bit = 1" in .emf with same "Table 1-6 Configuration Bit = 1" – Updated for Table 1-11 SENT I/O Control Selection and Setup aligned to TC23x V2.1 ITS/TS chapter 14 GPIO – Replaced on Table 1-5 Trigger Input Lines of SENT for TRIG3 from "GTM.DSADC_0_TRIG0" to "GTM.ADC_3_TRIG_0" – Added footnote on Table 1-5 Trigger Input Lines of SENT for TRIG 2 and TRIG 3 "These connections are not available on TC23x PD and are available on TC23x ED only, as TC23x PD has only ADC 0/1 while TC23x ED has ADC0/1/2/3."
Rev_1.34	Changed minimum tick timer to 0.2 µs For TC29x only: • Removed SENT2D from P40.6 • Moved SENT5A from P40.9 to P40.5 and SENT4A from P40.8 to P40.4 where available in product
Rev_1.34a	Updated Serial Message Frame Chapter

23 FlexRay™ Protocol Controller (E-Ray)

The E-Ray IP-module performs communication according to the FlexRay™ [1] protocol specification v2.1, developed for automotive applications. With maximum specified clock the bitrate can be programmed to values up to 10 Mbit/s. Additional bus driver (BD) hardware is required for connection to the physical layer.

23.1 E-Ray Kernel Description

Figure 23-1 shows a global view of the E-Ray interface.

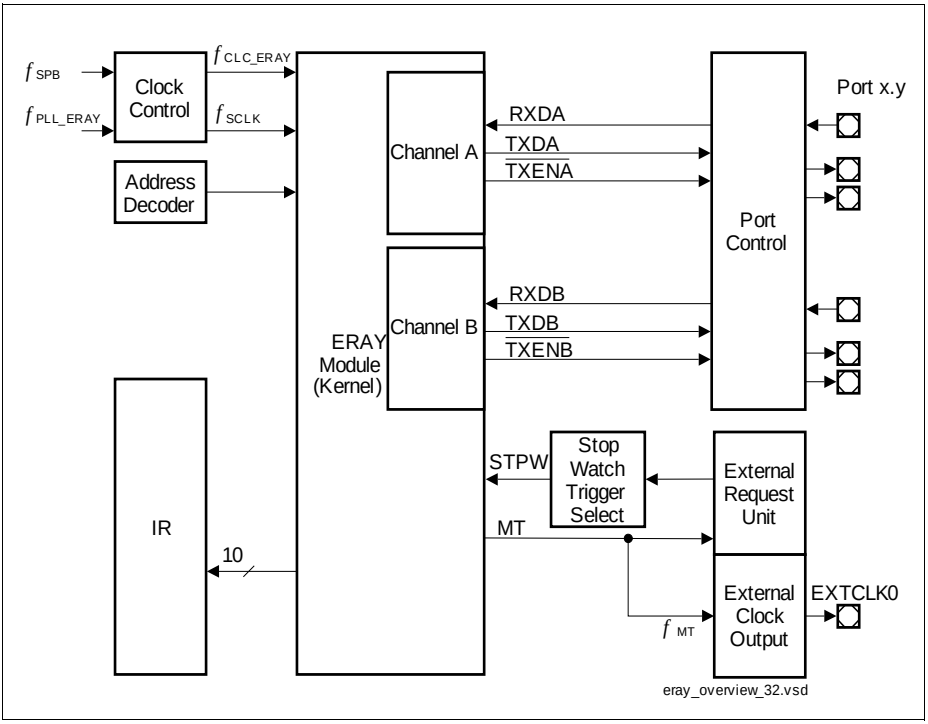


Figure 23-1 General Block Diagram of the E-Ray Interface

The E-Ray module communicates with the external world via three I/O lines each channel. The RXDAx and RXDBx lines are the receive data input signals, TXDA and

1) *Infineon®, Infineon Technologies®, are trademarks of Infineon Technologies AG. FlexRay™* is a trademark of FlexRay Consortium.

TXDB lines are the transmit output signals, $\overline{\text{TXENA}}$ and $\overline{\text{TXENB}}$ the transmit enable signals.

Clock control, address decoding, and service request control are managed outside the E-Ray module kernel.

23.2 Overview

For communication on a FlexRay™ network, individual Message Buffers with up to 254 data byte are configurable. The message storage consists of a single-ported Message RAM that holds up to 128 Message Buffers. All functions concerning the handling of messages are implemented in the Message Handler. Those functions are the acceptance filtering, the transfer of messages between the two FlexRay™ Channel Protocol Controllers and the Message RAM, maintaining the transmission schedule as well as providing message status information.

The register set of the E-Ray IP-module can be accessed directly by an external Host via the module's Host interface. These registers are used to control/configure/monitor the FlexRay™ Channel Protocol Controllers, Message Handler, Global Time Unit, System Universal Control, Frame and Symbol Processing, Network Management, Service Request Control, and to access the Message RAM via Input / Output Buffer.

The E-Ray IP-module supports the following features:

- Conformance with FlexRay™ protocol specification v2.1
- Data rates of up to 10 Mbit/s on each channel
- Up to 128 Message Buffers configurable
- 8 Kbyte of Message RAM for storage of e.g. 128 Message Buffers with max. 48 byte data field or up to 30 Message Buffers with 254 byte Data Sections
- Configuration of Message Buffers with different payload lengths possible
- One configurable receive FIFO
- Each Message Buffer can be configured as receive buffer, as transmit buffer or as part of the receive FIFO
- Host access to Message Buffers via Input and Output Buffer.
  Input Buffer: Holds message to be transferred to the Message RAM
  Output Buffer: Holds message read from the Message RAM
- Filtering for slot counter, cycle counter, and channel
- Maskable module service requests
- Network Management supported
- Four service request lines
- Automatic delayed read access to Output Command Request Register (OBCR) if a data transfer from Message RAM to Output Shadow Buffer (initiated by a previous write access to the OBCR) is ongoing.
- Automatic delayed read access to Input Command Request Register (IBCR) if a data transfer from Input Shadow Buffer to Message RAM to (initiated by a previous write access to the IBCR) is ongoing.

- Four Input Buffer for building up transmission Frames in parallel.
- Flag indicating which Input Buffer is currently accessible by the host.

23.3 Definitions

FlexRay™ Frame: Header Segment + Payload Segment

Message Buffer: Header Section + Data Section

Message RAM: Header Partition + Data Partition

Data Frame: FlexRay™ Frame that is not a NULL Frame

23.4 Block Diagram

The E-Ray is built up by the following main submodules:

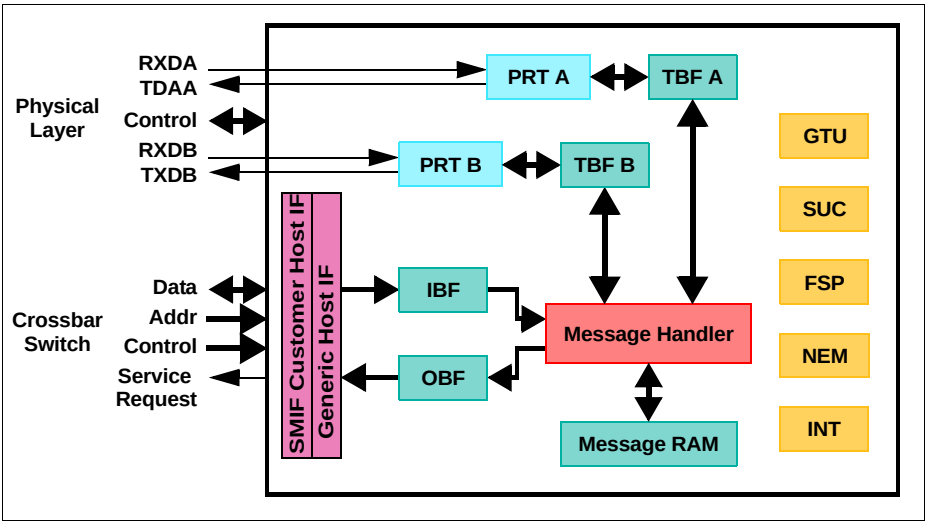


Figure 23-2 E-Ray Block Diagram

Customer Host Interface (CIF)

Connects the FPI Bus to the E-Ray IP-module via the Generic Host Interface.

Generic Host Interface (GIF)

The E-Ray IP-module is provided with an 8/16/32-bit Generic Host Interface prepared for the connection to a wide range of customer-specific Hosts. Configuration registers, status registers, and service request registers are attached to the respective blocks and can be accessed via the Generic Host Interface.

Input Buffer (IBF)

For write access to the Message Buffers configured in the Message RAM, the Host can write the Header and Data Section for a specific Message Buffer to the Input Buffer. The Message Handler then transfers the data from the Input Buffer to the selected Message Buffer in the Message RAM.

Because the Input Buffer (IBF) Scheme does only allow to write the entire Message Frame, not only parts of it, the number of IBF has been increased from originally 2 to 4. This enables to fill the buffer partly and at the end request transfer into Message RAM. Therefore 2 extra bits allow to switch between the two banks of IBF and one status bit signals the IBF currently active for Host writes.

Output Buffer (OBF)

For read access to a Message Buffer configured in the Message RAM the Message Handler transfers the selected Message Buffer to the Output Buffer. After the transfer has completed, the Host can read the Header and Data Section of the transferred Message Buffer from the Output Buffer.

Message Handler (MHD)

The E-Ray Message Handler controls data transfers between the following components:

- Input / Output Buffer and Message RAM
- Transient Buffer RAMs of the two FlexRay™ Protocol Controllers and Message RAM

Message RAM (MRAM)

The Message RAM consists of a single-ported RAM that stores up to 128 FlexRay™ Message Buffers together with the related configuration data (Header and Data Partition).

Transient Buffer RAM (TBF 1/2)

Stores the Data Section of two complete messages.

FlexRay™ Channel Protocol Controller (PRT A/B)

The FlexRay™ Channel Protocol Controllers consist of shift register and FlexRay™ protocol FSM. They are connected to the Transient Buffer RAMs for intermediate message storage and to the physical layer via bus driver BD.

They perform the following functionality:

- Control and check of bit timing
- Reception and transmission of FlexRay™ Frames and symbols
- Check of Header CRC
- Generation / check of Frame CRC

- Interfacing to bus driver

The FlexRay™ Channel Protocol Controllers have interfaces to:

- Physical Layer (bus driver)
- Transient Buffer RAM
- Message Handler
- Global Time Unit
- System Universal Control
- Frame and Symbol Processing
- Network Management
- Service Request Control

Global Time Unit (GTU)

The Global Time Unit performs the following functions:

- Generation of Microtick
- Generation of Macrotick
- Fault tolerant clock synchronization by FTM algorithm
  - Rate correction
  - Offset correction
- Cycle counter
- Timing control of static segment
- Timing control of dynamic segment (minislotting)
- Support of external clock correction

System Universal Control (SUC)

The System Universal Control controls the following functions:

- Configuration
- Wakeup
- Startup
- Normal Operation
- Passive Operation
- Monitor Mode

Frame and Symbol Processing (FSP)

The Frame and Symbol Processing controls the following functions:

- Checks the correct timing of Frames and symbols
- Tests the syntactical and semantical correctness of received Frames
- Sets the slot status flags

Network Management (NEM)

Handles of the Network Management vector

Service Request Control (INT)

The Service Request Controller performs the following functions:

- Provides error and status service request flags
- Enables and disables service request sources
- Assignment of service request sources to one of the two module service request lines
- Enables and disables module service request lines
- Manages the two service request timers
- Stop watch time capturing

23.5 Programmer's Model

The programmer's model of the E-Ray module follows the principle of memory mapped peripheral. Some portion of the memory follows the principle of segmented/paged memory organization.

23.5.1 Register Map

The E-Ray module allocates an address space of 4 Kbyte (000_H to $FFFF_H$). The registers are organized as 32-bit registers. 8/16-bit accesses are also supported. Host access to the Message RAM is done via the Input and Output Buffers. They buffer data to be transferred to and from the Message RAM under control of the Message Handler, avoiding conflicts between Host accesses and message reception / transmission. Addresses 0004_H - $000F_H$,$03C8_H$ - $03EC_H$ and 0800_H - $0FFF_H$ are reserved for customer specific purposes. All functions related to these addresses are located in the Customer Host Interface. The test registers located on address 0010_H and 0014_H are writable only under the conditions described in **"Special Registers" on Page 23-22**.

The assignment of the Message Buffers is done according to the scheme shown in **Table 23-1** below. The number N of available Message Buffers depends on the payload length of the configured Message Buffers. The maximum number of Message Buffers is 128. The maximum payload length supported is 254 byte.

The Message Buffers are separated into three consecutive groups:

- Static Buffers: Transmit / Receive Buffers assigned to static segment
- Static and Dynamic Buffers: Transmit / Receive Buffers assigned to static or dynamic segment
- FIFO- Receive FIFO

The Message Buffer separation configuration can be changed only in "DEFAULT_CONFIG" or "CONFIG" state only by programming the Message RAM Configuration register (MRC).

The first group starts with Message Buffer 0 and consists of static Message Buffers only. Message Buffer 0 is dedicated to hold the startup / SYNC Frame or the single slot Frame, if node transmit one, as configured by SUCC1.TXST, SUCC1.TXSY, and SUCC1.TSM

in the SUC Configuration Register 1 (SUCC1). In addition, Message Buffer 1 may be used for SYNC Frame transmission in case that SYNC Frames or single-slot Frames should have different payloads on the two channels. In this case bit MRC.SPLM has to be programmed to 1 and Message Buffers 0 and 1 have to be configured with the key slot ID and can be (re)configured in “DEFAULT_CONFIG” or “CONFIG” state only.

The second group consists of Message Buffers assigned to the static or to the dynamic segment. Message Buffers belonging to this group may be reconfigured during run time from dynamic to static or vice versa depending on the state of MRC.SEC.

The Message Buffers belonging to the third group are concatenated to a single receive FIFO.

Table 23-1 Assignment of Message Buffers

Message Buffer 0	⇓ Static Buffers	
Message Buffer 1		
…		
	⇓ Static + Dynamic Buffers	⇐ **FDB**
	⇓ FIFO	⇐ **FFB**
Message Buffer N-1		
Message Buffer N		⇐ **LCB**

23.5.2 E-Ray Kernel Registers

This chapter describes all registers of the E-Ray kernel.

All registers in the E-Ray address spaces are reset with the application reset with one exception: OCS is reset with debug reset only. (Definition see SCU section “Reset Operation”)

Table 23-2 Registers Address Space E-Ray Kernel Register Address Space

Module	Base Address	End Address	Note
ERAY0	$F001C000_H$	$F001CFFF_H$	4Kbyte

Table 23-3 Registers OverviewE-Ray Kernel Registers

Register Short Name	Register Long Name	Offset Addr.[1)]	Access Mode		Description see
			Read	Write	
Customer Registers					
CLC	Clock Control Register	0000_H	U, SV	SV, E	Page 23-262
CUST1	Busy and Input Buffer Control Register	0004_H	SV,U	SV,U	Page 23-16
ID	Module Identification Register	0008_H	SV,U	BE	Page 23-15
CUST3	Customer Interface Timeout Counter	$000C_H$	SV,U	SV,U	Page 23-19
Special Registers					
TEST1	Test Register 1	0010_H	SV,U	SV,U	Page 23-22
TEST2	Test Register 2	0014_H	SV,U	SV,U	Page 23-27
-	Reserved	0018_H	BE	BE	-
LCK	Lock Register	$001C_H$	SV,U	SV,U	Page 23-30
Service Request Registers					
EIR	Error Service Request Register	0020_H	SV,U	SV,U	Page 23-32
SIR	Status Service Request Register	0024_H	SV,U	SV,U	Page 23-38
EILS	Error Service Request Line Select	0028_H	SV,U	SV,U	Page 23-43
SILS	Status Service Request Line Select	$002C_H$	SV,U	SV,U	Page 23-47

Table 23-3 Registers OverviewE-Ray Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Addr.[1)]	Access Mode		Description see
			Read	Write	
EIES	Error Service Request Enable Set	0030_H	SV,U	SV,U	Page 23-51
EIER	Error Service Request Enable Reset	0034_H	SV,U	SV,U	Page 23-56
SIES	Status Service Request Enable Set	0038_H	SV,U	SV,U	Page 23-61
SIER	Status Service Request Enable Reset	$003C_H$	SV,U	SV,U	Page 23-66
ILE	Service Request Line Enable	0040_H	SV,U	SV,U	Page 23-71
T0C	Timer 0 Configuration	0044_H	SV,U	SV,U	Page 23-72
T1C	Timer 1 Configuration	0048_H	SV,U	SV,U	Page 23-74
STPW1	Stop Watch Register 1	$004C_H$	SV,U	SV,U	Page 23-76
STPW2	Stop Watch Register 2	0050_H	SV,U	SV,U	Page 23-78
-	Reserved	0054_H - $007C_H$	BE	BE	-
Communication Controller Control Registers					
SUCC1	SUC Configuration Register 1	0080_H	SV,U	SV,U	Page 23-79
SUCC2	SUC Configuration Register 2	0084_H	SV,U	SV,U	Page 23-87
SUCC3	SUC Configuration Register 3	0088_H	SV,U	SV,U	Page 23-88
NEMC	NEM Configuration Register	$008C_H$	SV,U	SV,U	Page 23-89
PRTC1	PRT Configuration Register 1	0090_H	SV,U	SV,U	Page 23-90
PRTC2	PRT Configuration Register 2	0094_H	SV,U	SV,U	Page 23-92
MHDC	MHD Configuration Register	0098_H	SV,U	SV,U	Page 23-93
-	Reserved	$009C_H$	BE	BE	-
GTUC01	GTU Configuration Register 1	$00A0_H$	SV,U	SV,U	Page 23-94
GTUC02	GTU Configuration Register 2	$00A4_H$	SV,U	SV,U	Page 23-95
GTUC03	GTU Configuration Register 3	$00A8_H$	SV,U	SV,U	Page 23-96
GTUC04	GTU Configuration Register 4	$00AC_H$	SV,U	SV,U	Page 23-97
GTUC05	GTU Configuration Register 5	$00B0_H$	SV,U	SV,U	Page 23-98
GTUC06	GTU Configuration Register 6	$00B4_H$	SV,U	SV,U	Page 23-99

Table 23-3 Registers OverviewE-Ray Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Addr.[1)]	Access Mode		Description see
			Read	Write	
GTUC07	GTU Configuration Register 7	$00B8_H$	SV,U	SV,U	Page 23-100
GTUC08	GTU Configuration Register 8	$00BC_H$	SV,U	SV,U	Page 23-101
GTUC09	GTU Configuration Register 9	$00C0_H$	SV,U	SV,U	Page 23-102
GTUC10	GTU Configuration Register 10	$00C4_H$	SV,U	SV,U	Page 23-103
GTUC11	GTU Configuration Register 11	$00C8_H$	SV,U	SV,U	Page 23-104
-	Reserved	$00CC_H$ - $00FC_H$	BE	BE	-
Communication Controller Status Registers					
CCSV	Communication Controller Status Vector	0100_H	SV,U	BE	Page 23-106
CCEV	Communication Controller Error Vector	0104_H	SV,U	BE	Page 23-111
-	Reserved	0108_H	nBE	BE	-
-	Reserved	$010C_H$	nBE	BE	-
SCV	Slot Counter Value	0110_H	SV,U	BE	Page 23-112
MTCCV	Macrotick and Cycle Counter Value	0114_H	SV,U	BE	Page 23-113
RCV	Rate Correction Value	0118_H	SV,U	BE	Page 23-114
OCV	Offset Correction Value	$011C_H$	SV,U	BE	Page 23-115
SFS	SYNC Frame Status	0120_H	SV,U	BE	Page 23-116
SWNIT	Symbol Window and Network Idle Time Status	0124_H	SV,U	BE	Page 23-118
ACS	Aggregated Channel Status	0128_H	SV,U	SV,U	Page 23-121
-	Reserved	$012C_H$	nBE	BE	-
ESIDnn	Even Sync ID Symbol Window nn	0130_H - 0168_H	SV,U	BE	Page 23-124
-	Reserved	$016C_H$	SV,U	BE	-
OSIDnn	Odd Sync ID Symbol Window nn	0170_H - $01A8_H$	SV,U	BE	Page 23-126
-	Reserved	$01AC_H$	SV,U	BE	-

Table 23-3 Registers OverviewE-Ray Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Addr.[1)]	Access Mode		Description see
			Read	Write	
NMVx	Network Management Vector [1…3]	$01B0_H$ - $01B8_H$	SV,U	BE	Page 23-128
-	Reserved	$01BC_H$ - $02FC_H$	nBE	BE	-
Message Buffer Control Registers					
MRC	Message RAM Configuration	0300_H	SV,U	SV,U	Page 23-129
FRF	FIFO Rejection Filter	0304_H	SV,U	SV,U	Page 23-132
FRFM	FIFO Rejection Filter Mask	0308_H	SV,U	SV,U	Page 23-134
FCL	FIFO Critical Level	$030C_H$	SV,U	SV,U	Page 23-135
Message Buffer Status Registers					
MHDS	Message Handler Status	0310_H	SV,U	SV,U	Page 23-136
LDTS	Last Dynamic Transmit Slot	0314_H	SV,U	SV,U	Page 23-138
FSR	FIFO Status Register	0318_H	SV,U	SV,U	Page 23-139
MHDF	Message Handler Constraints Flags	$031C_H$	SV,U	BE	Page 23-141
TXRQ1	Transmission Request Register 1	0320_H	SV,U	BE	Page 23-144
TXRQ2	Transmission Request Register 2	0324_H	SV,U	BE	Page 23-145
TXRQ3	Transmission Request Register 3	0328_H	SV,U	BE	Page 23-146
TXRQ4	Transmission Request Register 4	$032C_H$	SV,U	BE	Page 23-147
NDAT1	New Data Register 1	0330_H	SV,U	BE	Page 23-148
NDAT2	New Data Register 2	0334_H	SV,U	BE	Page 23-149
NDAT3	New Data Register 3	0338_H	SV,U	BE	Page 23-150
NDAT4	New Data Register 4	$033C_H$	SV,U	BE	Page 23-151
MBSC1	Message Buffer Status Changed 1	0340_H	SV,U	BE	Page 23-152
MBSC2	Message Buffer Status Changed 2	0344_H	SV,U	BE	Page 23-153
MBSC3	Message Buffer Status Changed 3	0348_H	SV,U	BE	Page 23-154

Table 23-3 Registers OverviewE-Ray Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Addr.[1)]	Access Mode		Description see
			Read	Write	
MBSC4	Message Buffer Status Changed 4	$034C_H$	SV,U	BE	Page 23-155
-	Reserved	0350_H - $03A4_H$	BE	BE	-
NDIC1	New Data Interrupt Control 1	$03A8_H$	SV,U	SV,U	Page 23-271
NDIC2	New Data Interrupt Control 2	$03AC_H$	SV,U	SV,U	Page 23-272
NDIC3	New Data Interrupt Control 3	$03B0_H$	SV,U	SV,U	Page 23-273
NDIC4	New Data Interrupt Control 4	$03B4_H$	SV,U	SV,U	Page 23-274
MSIC1	Message Buffer Status Changed Interrupt Control 1	$03B8_H$	SV,U	SV,U	Page 23-275
MSIC2	Message Buffer Status Changed Interrupt Control 2	$03BC_H$	SV,U	SV,U	Page 23-276
MSIC3	Message Buffer Status Changed Interrupt Control 3	$03C0_H$	SV,U	SV,U	Page 23-277
MSIC4	Message Buffer Status Changed Interrupt Control 4	$03C4_H$	SV,U	SV,U	Page 23-274
Identification Registers					
CREL	Core Release Registers	$03F0_H$	SV,U	nBE	Page 23-156
ENDN	Endian Register	$03F4_H$	SV,U	nBE	Page 23-158
-	Reserved	$03F6_H$ - $03FC_H$	BE	BE	-
Input Buffer					
WRDSn	Write Data Section [1…64]	0400_H - $04FC_H$	SV,U	SV,U	Page 23-159
WRHS1	Write Header Section 1	0500_H	SV,U	SV,U	Page 23-160
WRHS2	Write Header Section 2	0504_H	SV,U	SV,U	Page 23-163
WRHS3	Write Header Section 3	0508_H	SV,U	SV,U	Page 23-164
	Reserved	$050C_H$	BE	BE	
IBCM	Input Buffer Command Mask	0510_H	SV,U	SV,U	Page 23-165
IBCR	Input Buffer Command Request	0514_H	SV,U	SV,U	Page 23-167

Table 23-3 Registers OverviewE-Ray Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Addr.[1)]	Access Mode		Description see
			Read	Write	
	Reserved	0518_H - $05FC_H$	BE	BE	
Output Buffer					
RDDSn	Read Data Section [1…64]	0600_H - $06FC_H$	SV,U	BE	Page 23-169
RDHS1	Read Header Section 1	0700_H	SV,U	BE	Page 23-170
RDHS2	Read Header Section 2	0704_H	SV,U	BE	Page 23-172
RDHS3	Read Header Section 3	0708_H	SV,U	BE	Page 23-174
MBS	Message Buffer Status	$070C_H$	SV,U	BE	Page 23-176
OBCM	Output Buffer Command Mask	0710_H	SV,U	SV,U	Page 23-182
OBCR	Output Buffer Command Request	0714_H	SV,U	SV,U	Page 23-185
	Reserved	0718_H -	BE	BE	-
OTS Kernel Registers					
OTSS	OCDS Trigger Set Select	0870_H	U, SV	SV, E	Page 23-261
BPI Kernel Registers					
OCS	OCDS Control and Status Register	$08E8_H$	U, SV	SV, P	Page 23-264
KRSTCLR	Reset Status Clear Register	$08EC_H$	U, SV	SV, P	Page 23-269
KRST1	Reset Control Register 1	$08F0_H$	U, SV	SV, P	Page 23-268
KRST0	Reset Control Register 0	$08F4_H$	U, SV	SV, P	Page 23-267
ACCEN1	Access Enable Register 1	$08F8_H$	U, SV	SV, SE	Page 23-266
ACCEN0	Access Enable Register 0	$08FC_H$	U, SV	SV, SE	Page 23-265
	Reserved	0900_H - $0FFF_H$	BE	BE	-

1) The absolute register address is calculated as follows:
Module Base Address + Offset Address (shown in this column)

Note: Registers covered by the ACCENx write protection mechanism are marked with ´P´ in Access Mode, Write.

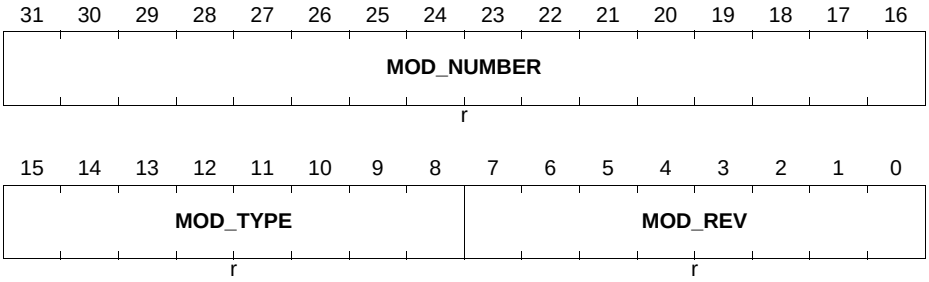
23.5.2.1 Customer Registers

The addresses 0004_H - $000F_H$,$03C8_H$ - $03EC_H$ and 0800_H - $0FFF_H$ are reserved for customer-specific registers.

Module Identification Register (ID)

This register contains bit fields identifying the E-Ray module in Infineons Module portfolio and is read only.

ID
Module Identification Register (0008_H) Reset Value: 0044 C0XX$_H$



Field	Bits	Type	Description
MOD_REV	[7:0]	r	**Module Revision Number** MOD_REV defines the module revision number. The value of a module revision starts with 01_H (first revision).
MOD_TYPE	[15:8]	r	**Module Type** The value of this bit field is $C0_H$. It defines the module as a 32-bit module.
MOD_NUMBER	[31:16]	r	**Module Number Value** This bit field defines a module identification number. For the E-Ray module the module identification number is 44_H.

Busy Control Register (CUST1)

The Busy Control Register enables the automatic delay scheme. Furthermore it signals a time-out service request for the automatic delay scheme. IBFS changes 2 clock cycles after any reset to ‘1’. Thus the application will read this bit as ‘1’ already at first access after any reset.

CUST1

Busy and Input Buffer Control Register

(0004_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STPWTS		RISB		RISA		0		IBF2 PAG	0		IBF1 PAG	IBFS	IEN	OEN	INT0
rw		rw		rw		r		rw	r		rw	rh	rw	rw	rwh

Field	Bits	Type	Description
INT0	0	rwh	**CIF Timeout Service Request Status** INT0 will be set if a timeout has occurred during the auto delay scheme and must be reset by writing zero to INT0. *Note: In case hardware sets INT0 and at the same point of time software clears INT0, INT0 is cleared.*
OEN	1	rw	**Enable auto delay scheme for Output Buffer Control Register (OBCR)** This control bit controls the delay scheme for Output Buffer Control Register (OBCR) read accesses. 0_B Disable auto delay scheme for Output Buffer Control Register (OBCR) 1_B Enable auto delay scheme for Output Buffer Control Register (OBCR)

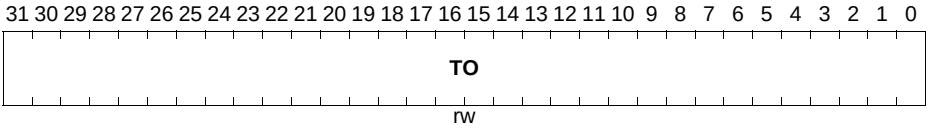
Field	Bits	Type	Description
IEN	2	rw	**Enable auto delay scheme for Input Buffer Control Register (IBCR)** This control bit controls the auto delay scheme for Input Buffer Control Register (IBCR) read accesses. 0_B Disable auto delay scheme for Input Buffer Control Register (IBCR) 1_B Enable auto delay scheme for Input Buffer Control Register (IBCR)
IBFS	3	rh	**Input Buffer Status Register** This status bit indicates which of the two Input Buffer RAMs (IBF) is accessible by the host (via CIF) as Input Buffer. The other non accessible buffer RAM is currently used as shadow buffer RAM by the ERAY message handler and therefore not accessible by the host.After reset, it is set by hardware. 0_B Input Buffer RAM 2 (IBF2) is accessible as Input Buffer by the host (CIF) 1_B Input Buffer RAM 1 (IBF1) is accessible as Input Buffer by the host (CIF)
IBF1PAG	4	rw	**Input Buffer 1 Page Select Register** This control bit selects if the upper page or lower page of Input Buffer 1 (IBF1) currently active. 0_B Read: Lower Page (256 Bytes) of Input Buffer RAM 1 selected Write: Select Lower Page (256 Bytes) of Input Buffer RAM 1 1_B Read: Upper Page (256 Bytes) of Input Buffer RAM 1 selected Write: Select Upper Page (256 Bytes) of Input Buffer RAM 1 *Note: Write is only possible, if Input Buffer RAM 1 is currently accessible by the host (via CIF) and therefore IBFS set.*

Field	Bits	Type	Description
IBF2PAG	7	rw	**Input Buffer 2 Page Select Register** This control bit selects if the upper page or lower page of Input Buffer 2 (IBF2) currently active. 0_B Read: Lower Page (256 Bytes) of Input Buffer RAM 2 selected Write: Select Lower Page (256 Bytes) of Input Buffer RAM 2 1_B Read: Upper Page (256 Bytes) of Input Buffer RAM 2 selected Write: Select Upper Page (256 Byte) of Input Buffer RAM 2 *Note: Write is only possible, if Input Buffer RAM 2 is currently accessible by the host (via CIF) and therefore IBFS cleared.*
RISA	[11:10]	rw	**Receive Input Select Channel A** 00_B Channel A receiver input RXDA0 selected 01_B Channel A receiver input RXDA1 selected 10_B Channel A receiver input RXDA2 selected 11_B Channel A receiver input RXDA3 selected
RISB	[13:12]	rw	**Receive Input Select Channel B** 00_B Channel B receiver input RXDB0 selected 01_B Channel B receiver input RXDB1 selected 10_B Channel B receiver input RXDB2 selected 11_B Channel B receiver input RXDB3 selected
STPWTS	[15:14]	rw	**Stop Watch Trigger Input Select** 00_B Stop Watch Trigger input STPWT0 selected 01_B Stop Watch Trigger input STPWT1 selected 10_B Stop Watch Trigger input STPWT2 selected 11_B Stop Watch Trigger input STPWT3 selected
0	[6:5], [9:8], [31:16]	r	**Reserved** Returns 0 if read; should be written with 0.

Customer Interface Time-out Counter Register (CUST3)

The Time-out Counter Register is realizing the time-out counter reload (startup) value for the automatic delay scheme (not the time-out down counter itself).

CUST3
Customer Interface Timeout Counter ($000C_H$) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
TO	[31:0]	rw	**CIF Timeout Reload Value** The 32-bit down counter reload (start-up) value must be setup for the automatic delay scheme.

Automatic Delayed Write Access to OBCR and IBCR

Write and read accesses to the Output Buffer Control Register (OBCR) can be automatically stalled due to a ongoing transfer from the Message Buffer to the Output Buffer. Also write and read accesses to the Input Buffer Control Register (IBCR) may be automatically delayed due to a ongoing transfer from the Input Buffer to the Message Buffer.

This delay scheme can be controlled (enabled or disabled) by CUST1.IEN and CUST1.OEN. The maximum time to stall a write or read access is determined by a single time-out counter preluded with the 32-bit value specified in the bit field CUST3.TO. If the time-out counter counts down to zero before the transfer to/from the Message Buffer is completed, the access (read or write) will be canceled and a service request will be generated. A canceled read access provides a 0 value. A canceled write access does not modify any bits in the OBCR or IBCR. In addition the bit CUST1.INT0 of the service request status register will be set and must be reset by the host to disable the service request line.

The read and write access to the Output Buffer Control Register (OBCR) may be configured without automatic delay by clearing CUST1.OEN. Setting OBCR.REQ and immediately afterwards reading or writing OBCR, e.g. to set OBCR.VIEW will lead to a canceled read or write operation, e.g. OBCR.VIEW remains cleared, and an error is signalled by a set EIR.IOBA. Besides canceling the erroneous read or write operation, and setting the error bit, no further state change happens. So full operation is granted. OBCR remains read and write inaccessible until the transfer of data from the Message Buffer to the Output Buffer (MBF⇒OBF) is completed. During this time span all read and

write accesses to the Output Buffer Control Register (OBCR) are canceled. The transfer is completed when OBCR.OBSYS is cleared. Additionally signal TOBC may be used, e.g. for service request triggering, DMA triggering, or driving a pin, to communicate the access status.

The read and write access to the Output Buffer Control Register (OBCR) may be configured to be automatic delayed by setting CUST1.OEN and configuring CUST3.TO to the maximum stall time acceptable to the system. If setting OBCR.REQ and immediately afterwards reading or writing to OBCR, e.g. to set the OBCR.VIEW bit, this read or write will be stalled until either the maximum delay time elapsed (in this case the read or write operation is cancelled after the stall time, e.g. OBCR.VIEW remains cleared, and an error is signalled by setting EIR.IOBA) or the read or write completes normally, e.g. set OBCR.VIEW after the transfer of data from the Message Buffer to the Output Buffer (MBF⇒OBF) is finalized. During this time the bus is locked and no further access to the E-Ray module is possible due to the ongoing stalled read or write operation. Because no access is possible to the E-Ray module, read or write stall may only be detected through the signal TOBC or due to other not processed read or write accesses to the E-Ray module.

The read and write access to the Input Buffer Control Register (IBCR) may also be configured without automatic delay by clearing CUST1.IEN. By writing to IBCR.IBRH the Input Buffers are swapped (shadow IBF changes to host IBF and host IBF to shadow IBF), the content of the shadow IBF is copied into the MBF (IBF⇒MBF), and IBCR.IBSYS is set. Writing to IBCR.IBRH a second time while IBCR.IBSYS remained set (previously initiated copy process IBF⇒MBF ongoing) will correctly update IBCR.IBRH and set IBCR.IBSYH. This will set the signal IBUSY. A third access, read or write, to IBCR while IBCR.IBSYH remains set will cancel this third access and an error is signalled by setting EIR.IIBA. Besides canceling this last access to IBCR and setting the error bit, no further state change happens. So full operation is granted. IBCR remains read and write inaccessible until the transfer of data from the Input Shadow Buffer to the Message Buffer (IBF⇒MBF) is completed and once more the Input Buffers are swapped (shadow IBF changes to host IBF and host IBF to shadow IBF). During this time span all read and write accesses to the Input Buffer Control Register (IBCR) are canceled. The transfer is completed when IBCR.IBSYH is cleared. Additionally signal TIBC may be used, e.g. for service request triggering, DMA triggering, or driving a pin, to communicate the access status.

The read and write access to the Input Buffer Control Register (IBCR) may be configured for being automatically delayed by setting CUST1.IEN and configuring CUST3.TO to the maximum stall time acceptable to the system. By writing to IBCR.IBRH the Input Buffers are swapped (shadow IBF changes to host IBF and host IBF to shadow IBF), the content of the shadow IBF copied into the MBF, and IBCR.IBSYS is set. Writing to IBCR.IBRH a second time while IBCR.IBSYS remains set (previously initiated copy process ongoing) will correctly update IBCR.IBRH and set IBCR.IBSYH. A third access to IBCR while IBCR.IBSYH remains set will stall this read or write until either the maximum delay

time elapsed (in this case the read or write operation is cancelled after the stall time and an error is signalled by setting EIR.IOBA) or the read or write completes normally, after the transfer of data from the Input Shadow Buffer to the Message Buffer (IBF⇒MBF) is finalized and once more the Input Buffers are swapped (shadow IBF changes to host IBF and host IBF to shadow IBF). During this time the bus is locked and no further access to E-Ray module is possible due to the ongoing stalled read or write operation. Because no access is possible to the E-Ray module, read or write stall may only be detected through the signal TIBC or due to other not processed read or write accesses to the E-Ray module.

So setting CUST3.TO = $FFFFFFFF_H$, CUST1.IEN = 1, and CUST1.OEN = 1 will always grant a consistent data access of the host to the Output and Input Buffers without the need of reading and taking into account the status of OBCR.OBSYS or IBCR.IBSYH. But this simplified access may cause system latencies and system performance loss.

23.5.2.2 Special Registers

Test Register 1 (TEST1)

The Test Register 1 holds the control bits to configure the test modes of the E-Ray module. Write access to these bits is only possible if bit TEST1.WRTEN is set.

The Test Register 1 bits therefore can be used to test the interface to the physical layer (connectivity test) by driving / reading the respective pins.

When the E-Ray IP is operated in one of its test modes that requires TEST1.WRTEN to be set (RAM Test Mode, I/O Test Mode, Asynchronous Transmit Mode, and Loop Back Mode) only the selected test mode functionality is available.

The test functions are not available in addition to the normal operational mode functions, they change the functions of parts of the E-Ray module. Therefore normal operation as specified outside this chapter and as required by the FlexRay™ protocol specification and the FlexRay™ conformance test is not possible. Test mode functions may not be combined with each other or with FlexRay™ protocol functions.

The test mode features are intended for hardware testing or for FlexRay™ bus analyzer tools. They are not intended to be used in FlexRay™ applications

TEST1
Test Register 1 **(0010_H)** **Reset Value: $0000\ 0300_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CERB				CERA				0		TXENB	TXENA	TXB	TXA	RXB	RXA
rh				rh				r		rwh	rwh	rwh	rwh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						AOB	AOA	0		TMC		0		ELBE	WRTEN
r						rh	rh	r		rw		r		rw	rw

Field	Bits	Type	Description
WRTEN	0	rw	**Write Test Register Enable** Enables write access to the test registers. To set the bit from 0 to 1 the test mode key has to be written as defined on **"Lock Register (LCK)" on Page 23-30**. The unlock sequence is not required when TEST1.WRTEN is kept at 1 while other bits of the register are changed. The bit can be reset to 0 at any time. 0_B Write access to test registers disabled. 1_B Write access to test registers enabled.

Field	Bits	Type	Description
ELBE	1	rw	**External Loop Back Enable** There are two possibilities to perform a loop back test. External loop back via physical layer or internal loop back for in-system self-test (default). In case of an internal loop back pins TXENA and TXENB are in their inactive state, pins TXDA and TXDB are set to HIGH, pins RXDA and RXDB are not evaluated. Bit ELBE is evaluated only when POC is in loop back mode and test multiplexer control is in non multiplexed mode TMC = 00. 0_B Internal loop back (default) 1_B External loop back
TMC	[5:4]	rw	**Test Multiplexer Control** 00_B Normal signal path (default). 01_B RAM Test Mode: Internal busses are multiplexed to make all RAM blocks of the E-Ray module directly accessible by the Host. This mode is intended to enable testing of the embedded RAM blocks during production testing. 10_B I/O Test Mode: Output pins are driven to the values defined by bits TXA, TXB, TXENA, TXENB. The values applied to the input pins can be read from register bits RXA and RXB. 11_B Reserved; should not be used.
AOA	8	rh	**Activity on A** The channel idle condition is specified in the FlexRay™ protocol spec v2.1, chapter 3, BITSTRB process (zChannelIdle). 0_B No activity detected, channel A idle 1_B Activity detected, channel A not idle
AOB	9	rh	**Activity on B** The channel idle condition is specified in the FlexRay™ protocol spec v2.1, chapter 3, BITSTRB process (zChannelIdle). 0_B No activity detected, channel B idle 1_B Activity detected, channel B not idle
RXA	16	rh	**Read Channel A Receive Pin** 0_B RXDA = 0 1_B RXDA = 1

Field	Bits	Type	Description
RXB	17	rh	**Read Channel B Receive Pin** 0_B RXDB = 0 1_B RXDB = 1
TXA	18	rwh	**Read or Write to Channel A Transmit Pin** 0_B TXDA = 0 1_B TXDA = 1
TXB	19	rwh	**Read or Write to Channel B Transmit Pin** 0_B TXDB = 0 1_B TXDB = 1
TXENA	20	rwh	**Read or Write to Channel A Transmit Enable Pin** 0_B $\overline{\text{TXENA}}$= 0 1_B $\overline{\text{TXENA}}$= 1
TXENB	21	rwh	**Read or Write to Channel B Transmit Enable Pin** 0_B $\overline{\text{TXENB}}$ = 0 1_B $\overline{\text{TXENB}}$= 1
CERA	[27:24]	rh	**Coding Error Report Channel A[1)]** Set when a coding error is detected on channel A. Reset to zero when register TEST1 is read or written. Once the CERA is set it will remain unchanged until the Host accesses the TEST1 register. 0000_B No coding error detected 0001_B Header CRC error detected 0010_B Frame CRC error detected 0011_B Frame Start Sequence FSS too long 0100_B First bit of Byte Start Sequence BSS seen LOW 0101_B Second bit of Byte Start Sequence BSS seen HIGH 0110_B First bit of Frame End Sequence FES seen HIGH 0111_B Second bit of Frame End Sequence FES seen LOW 1000_B CAS / MTS symbol seen too short 1001_B CAS / MTS symbol seen too long Other combinations are reserved.

Field	Bits	Type	Description
CERB	[31:28]	rh	**Coding Error Report Channel B[1)]** Set when a coding error is detected on channel B. Reset to zero when register TEST1 is read or written. Once the CERB is set it will remain unchanged until the Host accesses the TEST1 register. 0000_B No coding error detected 0001_B Header CRC error detected 0010_B Frame CRC error detected 0011_B Frame Start Sequence FSS too long 0100_B First bit of Byte Start Sequence BSS seen LOW 0101_B Second bit of Byte Start Sequence BSS seen HIGH 0110_B First bit of Frame End Sequence FES seen HIGH 0111_B Second bit of Frame End Sequence FES seen LOW 1000_B CAS / MTS symbol seen too short 1001_B CAS / MTS symbol seen too long Other combinations are reserved.
0	[3:2], [7:6], [15:10], [23:22]	r	**Reserved** Returns 0 if read; should be written with 0.

1) Coding errors are also signalled when the Communication Controller is in “MONITOR_MODE”. The error codes regarding CAS / MTS symbols concern only the monitored bit pattern, irrelevant whether those bit patterns are seen in the symbol window or elsewhere.

Asynchronous Transmit Mode (ATM)

The asynchronous transmit mode is entered by writing 1110_B to the CHI Command Vector SUCC1.CMD in the SUC Configuration Register 1 (CHI command: ATM) while the Communication Controller is in “CONFIG” state and bit TEST1.WRTEN in the Test Register 1 is set. This write operation has to be directly preceded by two consecutive write accesses to the Configuration Lock Key (unlock sequence). When called in any other state or when bit TEST1.WRTEN is not set, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”. CCSV.POCS in the Communication Controller Status Vector will return 1110_B while the E-Ray module is in ATM mode. Asynchronous Transmit mode can be left by writing 0001_B (CHI command: “CONFIG”) to the CHI Command Vector SUCC1.CMD in the SUC Configuration Register 1.

In ATM mode transmission of a FlexRay™ Frame is triggered by writing the number of the respective Message Buffer to the Input Buffer Command Request register (IBCR.IBRH) while bit IBCM.STXRS in the Input Buffer Command Mask register is set to 1. In this mode wake-up, startup, and clock synchronization are bypassed. The CHI command SEND_MTS results in the immediate transmission of an MTS symbol.

The cycle counter value of Frames send in ATM mode can be programmed via MTCCV.CCV (writable in ATM and loop back mode only).

Loop Back Mode

The loop back mode is entered by writing 1111_B to the CHI Command Vector SUCC1.CMD in the SUC Configuration Register 1 (CHI command: LOOP_BACK) while the Communication Controller is in “CONFIG” state and bit TEST1.WRTEN in the Test Register 1 is set. This write operation has to be directly preceded by two consecutive write accesses to the Configuration Lock Key (unlock sequence). When called in any other state or when bit TEST1.WRTEN is not set, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”. CCSV.POCS in the Communication Controller Status Vector will show 0000 1101_H while the E-Ray module is in loop back mode.

Loop Back mode can be left by writing 0001_B (CHI command: “CONFIG”) to the CHI Command Vector SUCC1.CMD in the SUC Configuration Register 1.

The loop back test mode is intended to check the module’s internal data paths. Normal, time triggered operation is not possible in loop back mode.

There are two possibilities to perform a loop back test. External loop back via physical layer (TEST1.ELBE = 1) or internal loop back for in-system self-test (TEST1.ELBE = 0). In case of an internal loop back pins $\overline{\text{TXENA}}$, $\overline{\text{TXENB}}$ are in their inactive state, pins TXDA and TXDB are set to HIGH, pins RXDAn and RXDBn are not evaluated.

When the Communication Controller is in loop back mode, a loop back test is started by the Host writing a message to the Input Buffer and requesting the transmission by writing to the Input Buffer Command Request register IBCR. The Message Handler will transfer the message into the Message RAM and then into the Transient Buffer of the selected channel. The Channel Protocol Controller (PRT) will read (in 32-bit words) the message from the transmit part of the Transient Buffer and load it into its Rx / Tx shift register. The serial transmission is looped back into the shift register; its content is written into the receive part of the channels’s Transient Buffer before the next word is loaded.

The PRT and the Message Handler will then treat this transmitted message like a received message, perform an acceptance filtering on Frame ID and receive channel, and store the message into the Message RAM if it passed acceptance filtering. The loop back test ends with the Host requesting this received message from the Message RAM and then checking the contents of the Output Buffer.

Each FlexRay™ channel is tested separately. The E-Ray cannot receive messages from the FlexRay™ bus while it is in the loop back mode.

The cycle counter value of Frames used in loop back mode can be programmed via MTCCV.CCV (writable in ATM and loop back mode only).

Note that in case of an odd payload the last two bytes of the looped-back payload will be shifted by 16 bits to the right inside the last 32-bit data word.

Test Register 2 (TEST2)

The Test Register 2 holds all bits required for the RAM test of the seven embedded RAM blocks of the E-Ray module. Write access to this register is only possible when TEST1.WRTEN in the Test Register 1 is set to 1.

TEST2
Test Register 2 **(0014_H)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	WR ECC	0							SSEL			0	RS		
r	rw	r							rw			r	rw		

Field	Bits	Type	Description
RS	[2:0]	rw	**RAM Select** In RAM Test mode the RAM blocks selected by RS are mapped to module address $0000\ 0400_H$ to $0000\ 07FF_H$ (1024 byte addresses). 000_B Input Buffer RAM 1 (IBF1) 001_B Input Buffer RAM 2 (IBF2) 010_B Output Buffer RAM 1 (OBF1) 011_B Output Buffer RAM 2 (OBF2) 100_B Transient Buffer RAM A (TBF1) 101_B Transient Buffer RAM B (TBF2) 110_B Message RAM (MBF) 111_B Reserved; should not be used.

Field	Bits	Type	Description
SSEL	[6:4]	rw	**Segment Select** To enable access to the complete Message RAM (8192 byte addresses) the Message RAM is segmented. 000_B access to RAM byte 0000_H to $03FF_H$ enabled 001_B access to RAM byte 0400_H to $07FF_H$ enabled 010_B access to RAM byte 0800_H to $0BFF_H$ enabled 011_B access to RAM byte $0C00_H$ to $0FFF_H$ enabled 100_B access to RAM byte 1000_H to $11FF_H$ enabled 101_B access to RAM byte 1400_H to $17FF_H$ enabled 110_B access to RAM byte 1800_H to $1BFF_H$ enabled 111_B access to RAM byte $1C00_H$ to $1FFF_H$ enabled
WR ECC	14	rw	**Write ECC Data Enable** Content of ECCW is transferred to the RAM: 0_B disabled 1_B enabled *Note: Test mode must be entered. See* ***"Test Register 1 (TEST1)" on Page 23-22***
0	3, [13:7], 15, [31:16]	r	**Reserved** Returns 0 if read; should be written with 0.

RAM Test Mode

In RAM test mode (TEST1.TMC = 1), one of the seven RAM blocks can be selected for direct RD/WR access by programming TEST2.RS.

For external access the selected RAM block is mapped to address space 400_H to $7FF_H$ (1024 byte addresses or 256 word addresses).

Because the length of the Message RAM exceeds the available address space, the Message RAM is segmented into segments of 1024 byte. The segments can be selected by programming TEST2.SSEL in the Test Register 2.

In RAM test mode the memory must be accessed with 32Bit accesses only.

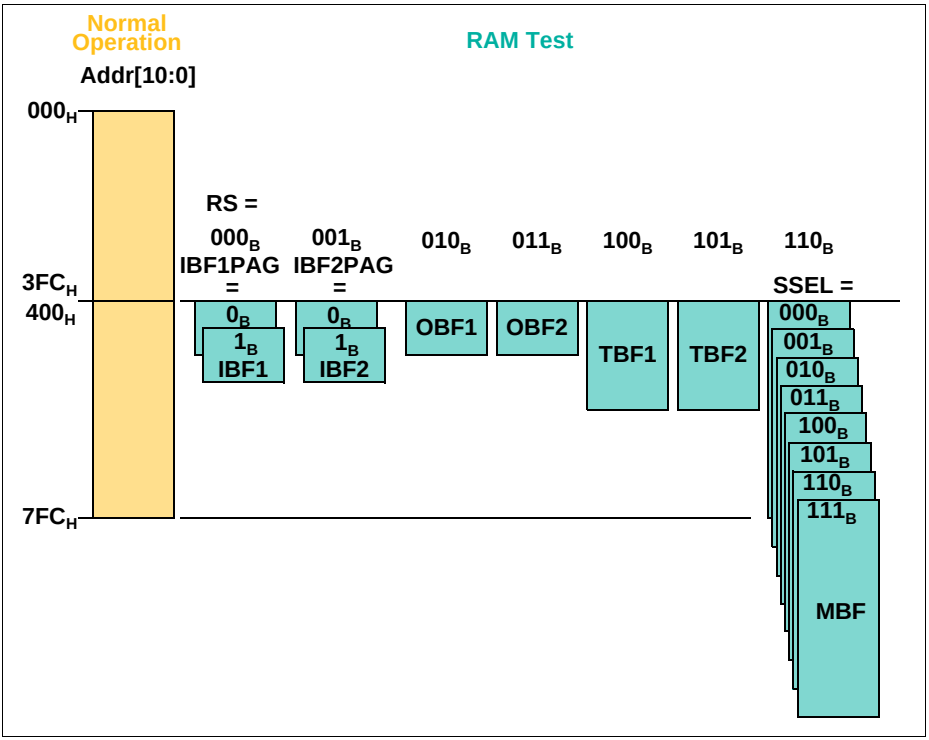
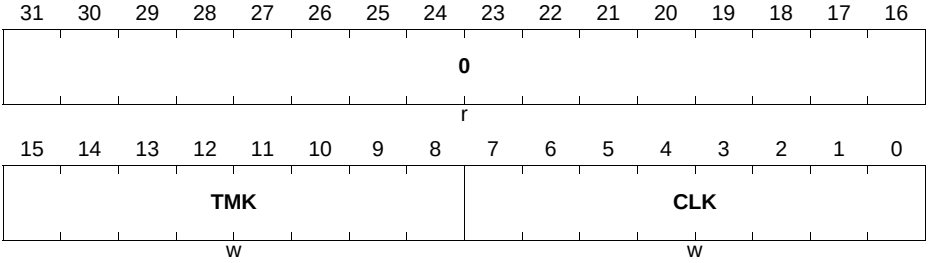


Figure 23-3 RAM test mode Access to E-Ray RAM Blocks

Lock Register (LCK)

The Lock Register is write-only. Reading the register will return 0000 0000_H.

LCK
Lock Register **($001C_H$)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
CLK	[7:0]	w	**Configuration Lock Key** To leave “CONFIG” state by writing to SUCC1.CMD commands READY, MONITOR_MODE, ATM, LOOP_BACK) in the SUC Configuration Register 1, the write operation has to be directly preceded by two consecutive write accesses to the Configuration Lock Key (unlock sequence). If the write sequence below is interrupted by other write accesses between the second write to the Configuration Lock Key and the write access to the SUCC1 register, the Communication Controller remains in “CONFIG” state and the sequence has to be repeated. First write: LCK.CLK = CE_H = 1100 1110_B Second write: LCK.CLK = 31_H = 0011 0001_B Third write: SUCC1.CMD Returns 0 if read

Field	Bits	Type	Description
TMK	[15:8]	w	**Test Mode Key** To set bit TEST1.WRTEN the write operation has to be directly preceded by two consecutive write accesses to the Test Mode Key. If the write sequence is interrupted by other write accesses between the second write to the Test Mode Key and the write access to the Test1 register, bit TEST1.WRTEN is not set to 1 and the sequence has to be repeated. First write: LCK.TMK = 75_H = 0111 0101_B Second write: LCK.TMK = $8A_H$ = 1000 1010_B Second write: TEST1.WRTEN = 1 Returns 0 if read
0	[31:16]	r	**Reserved** Returns 0 if read; should be written with 0.

Note: In case the Host uses 8/16-bit accesses to write the listed bit fields, the programmer has to ensure that no "dummy accesses" e.g. to the remaining register bytes / words are inserted by the compiler.

To exit "CONFIG" state by writing to SUCC1.CMD in the SUC Configuration Register 1, the write operation has to be directly preceded by two consecutive write accesses to the Configuration Lock Key. If this write sequence is service requested by read accesses or write accesses to other locations, the Communication Controller remains in "CONFIG" state and the sequence has to be repeated.

First write: LCK.CLK = CE_H = 1100 1110_B

Second write: LCK.CLK = 31_H = 0011 0001_B

23.5.2.3 Service Request Registers

The address space from 0020_H to $007F_H$ is reserved for service request registers.

Error Service Request Select (EIR)

The flags are set when the Communication Controller detects one of the listed error conditions. They remain set until the Host clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect on the flag. An application reset will also clear the register.

EIR
Error Service Request Register (0020_H) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					TABB	LTVB	EDB	0					TABA	LTVA	EDA
r					rwh	rwh	rwh	r					rwh	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				MHF	IOBA	IIBA	EFA	RFO	EERR	CCL	CCF	SFO	SFBM	CNA	PEMC
r				rwh	rwh	rwh	rwh	rh	rh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
PEMC	0	rwh	**POC Error Mode Changed** This flag is set whenever the error mode signalled by CCEV.ERRM in the Communication Controller Error Vector register has changed. 0_B Error mode has not changed 1_B Error mode has changed This flag is cleared by writing a 1.
CNA	1	rwh	**Command Not Accepted** The flag signals that the write access to the CHI command vector SUCC1.CMD in the SUC Configuration Register 1 was not successful because the requested command was not valid in the actual POC state, or because the CHI command was locked (CCL = 1). 0_B CHI command accepted 1_B CHI command not accepted This flag is cleared by writing a 1.

Field	Bits	Type	Description
SFBM	2	rwh	**SYNC Frames Below Minimum** This flag signals that the number of SYNC Frames received during the last communication cycle was below the limit required by the FlexRay™ protocol. May be set during startup and therefore should be cleared by the Host after the Communication Controller entered “NORMAL_ACTIVE” state. 0_B Sync node: 1 or more SYNC Frames received Non-sync node: 2 or more SYNC Frames received 1_B Less than the required minimum of SYNC Frames received This flag is cleared by writing a 1.
SFO	3	rwh	**SYNC Frame Overflow** Set when either the number of SYNC Frames received during the last communication cycle or the total number of SYNC Frames received during the last double cycle exceeds the maximum number of SYNC Frames as defined by GTUC02.SNM in the GTU Configuration Register 2. 0_B Number of received SYNC Frames ≤ GTUC02.SNM 1_B More SYNC Frames received than configured by GTUC02.SNM This flag is cleared by writing a 1.
CCF	4	rwh	**Clock Correction Failure** This flag is set at the end of the cycle whenever one of the following errors occurred: • Missing offset and / or rate correction • Clock Correction limit reached The clock correction status is monitored in registers CCEV and SFS. A failure may occur during startup, therefore bit CCF should be cleared by the Host after the Communication Controller entered “NORMAL_ACTIVE” state. 0_B Clock correction successful so far 1_B Clock correction failed This flag is cleared by writing a 1.

Field	Bits	Type	Description
CCL	5	rwh	**CHI Command Locked** The flag signals that the write access to the CHI command vector SUCC1.CMD was not successful because the execution of the previous CHI command has not yet completed. In this case bit EIR.CNA is also set to 1. 0_B CHI command accepted 1_B CHI command not accepted This flag is cleared by writing a 1.
EERR	6	rh	**ECC Error** The flag signals an ECC error to the Host. It is set whenever one of the flags MHDS.EIBF, MHDS.EOBF, MHDS.EMR, MHDS.ETBF1, MHDS.ETBF2 changes from 0 to 1. See also **“Message Handler Status (MHDS)” on Page 23-136**. This bit must be cleared at initialization of the module! 0_B No error detected 1_B Error detected
RFO	7	rh	**Receive FIFO Overrun** The flag is set by the Communication Controller when a receive FIFO overrun is detected. When a receive FIFO overrun occurs, the oldest message is overwritten with the actual received message. The actual state of the FIFO is monitored in register FSR. 0_B No receive FIFO overrun detected 1_B A receive FIFO overrun has been detected
EFA	8	rwh	**Empty FIFO Access** This flag is set by the Communication Controller when the Host requests the transfer of a message from the receive FIFO via Output Buffer while the receive FIFO is empty. 0_B No Host access to empty FIFO occurred 1_B Host access to empty FIFO occurred

Field	Bits	Type	Description
IIBA	9	rwh	**Illegal Input Buffer Access** This flag is set by the Communication Controller when the Host wants to modify a Message Buffer via Input Buffer while the Communication Controller is not in “CONFIG” or “DEFAULT_CONFIG” state and one of the following conditions applies: 1. The Host writes to the Input Buffer Command Request register to modify the: a) Header Section of Message Buffer 0, 1 if configured for transmission in key slot b) Header Section of static Message Buffers with buffer number < MRC.FDB while MRC.SEC = 01_B c) Header Section of any static or dynamic Message Buffer while MRC.SEC = $1x_B$ d) Header and / or Data Section of any message buffer belonging to the receive FIFO 2. The Host writes to any register of the Input Buffer while IBCR.IBSYS is set. 0_B No illegal Host access to Input Buffer occurred 1_B Illegal Host access to Input Buffer occurred
IOBA	10	rwh	**Illegal Output Buffer Access** This flag is set by the Communication Controller when the Host requests the transfer of a Message Buffer from the Message RAM to the Output Buffer while OBCR.OBSYS is set to 1. 0_B No illegal Host access to Output Buffer occurred 1_B Illegal Host access to Output Buffer occurred
MHF	11	rwh	**Message Handler Constraints Flag** The flag signals a Message Handler constraints violation condition. It is set whenever one of the flags MHDF.SNUA, MHDF.SNUB, MHDF.FNFA, MHDF.FNFB, MHDF.TBFA, MHDF.TBFB, MHDF.TNSA, MHDF.TNSB, MHDF.WAHP changes from 0 to 1. 0_B No Message Handler failure detected 1_B Message Handler failure detected

Field	Bits	Type	Description
EDA	16	rwh	**Error Detected on Channel A** This bit is set whenever one of the flags ACS.SEDA, ACS.CEDA, ACS.CIA, ACS.SBVA changes from 0 to 1. 0_B No error detected on channel A 1_B Error detected on channel A This flag is cleared by writing a 1.
LTVA	17	rwh	**Latest Transmit Violation Channel A** The flag signals a latest transmit violation on channel A to the Host. 0_B No latest transmit violation detected on channel A 1_B Latest transmit violation detected on channel A This flag is cleared by writing a 1.
TABA	18	rwh	**Transmission Across Boundary Channel A** The flag signals to the Host that a transmission across a slot boundary occurred for channel A. 0_B No transmission across slot boundary detected on channel A 1_B Transmission across slot boundary detected on channel A This flag is cleared by writing a 1.
EDB	24	rwh	**Error Detected on Channel B** This bit is set whenever one of the flags ACS.SEDB, ACS.CEDB, ACS.CIB, ACS.SBVB changes from 0 to 1. 0_B No error detected on channel B 1_B Error detected on channel B This flag is cleared by writing a 1.
LTVB	25	rwh	**Latest Transmit Violation Channel B** The flag signals a latest transmit violation on channel B to the Host. 0_B No latest transmit violation detected on channel B 1_B Latest transmit violation detected on channel B This flag is cleared by writing a 1.

Field	Bits	Type	Description
TABB	26	rwh	**Transmission Across Boundary Channel B** The flag signals to the Host that a transmission across a slot boundary occurred for channel B. 0_B No transmission across slot boundary detected on channel B 1_B Transmission across slot boundary detected on channel B This flag is cleared by writing a 1.
0	[15:12], [23:19], [31:27]	r	**Reserved** Returns 0 if read; should be written with 0.

Status Service Request Register (SIR)

The flags are set whenever the Communication Controller detects one of the listed events. The flags remain set until the Host clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect on the flag. An application reset will also clear the register.

SIR
Status Service Request Register (0024$_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						MTSB	WUPB	0						MTSA	WUPA
r						rwh	rwh	r						rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDS	MBSI	SUCS	SWE	TOBC	TIBC	TI1	TI0	NMVC	RFCL	RFNE	RXI	TXI	CYCS	CAS	WST
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rh	rh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
WST	0	rwh	**Wakeup Status** This flag is set when the wakeup status vector CCSV.WSV in the Communication Controller Status Vector register changes to a value other than UNDEFINED. 0_B Wake-up status unmodified 1_B Wake-up status modified (and not UNDEFINED) This flag is cleared by writing a 1.
CAS	1	rwh	**Collision Avoidance Symbol** This flag is set by the Communication Controller during STARTUP state when a CAS or potential CAS was received. 0_B No bit pattern matching the CAS symbol received 1_B Bit pattern matching the CAS symbol received This flag is cleared by writing a 1.
CYCS	2	rwh	**Cycle Start Service Request** This flag is set by the Communication Controller when a communication cycle starts 0_B No communication cycle started 1_B Communication cycle started This flag is cleared by writing a 1.

Field	Bits	Type	Description
TXI	3	rwh	**Transmit Service Request** This flag is set by the Communication Controller at the end of Frame transmission if bit WRHS1.MBI in the respective Message Buffer is set (see **Table 23-24**). 0_B No Frame transmitted from a transmit buffer with WRHS1.MBI = 1 1_B At least one Frame was transmitted from a transmit buffer with WRHS1.MBI = 1 This flag is cleared by writing a 1.
RXI	4	rwh	**Receive Service Request** This flag is set by the Communication Controller whenever the set condition of a Message Buffer ND flag is fulfilled and if bit WRHS1.MBI of that Message Buffer is set to 1(see **Table 23-24**). 0_B No ND flag of a receive buffer with WRHS1.MBI = 1 has been set to 1 1_B At least one ND flag of a receive buffer with WRHS1.MBI = 1 has been set to 1 This flag is cleared by writing a 1.
RFNE	5	rh	**Receive FIFO Not Empty** This flag is set by the Communication Controller when a received valid Frame was stored into the empty receive FIFO.m The actual state of the receive FIFO is monitored in register FSR 0_B Receive FIFO is empty 1_B Receive FIFO is not empty
RFCL	6	rh	**Receive FIFO Critical Level** This flag is set when a valid receive FIFO fill level FSR.RFFL is equal or greater than the critical level as configured by FCL.CL. 0_B Receive FIFO below critical level 1_B Receive FIFO critical level reached
NMVC	7	rwh	**Network Management Vector Changed** This service request flag signals a change in the Network Management Vector visible to the Host. 0_B No change in the Network Management vector 1_B Network Management vector changed This flag is cleared by writing a 1.

Field	Bits	Type	Description
TI0	8	rwh	**Timer Service Request 0** This flag is set whenever timer 0 matches the conditions configured in the Timer Service Request 0 Configuration Register T0C. A Timer Service Request 0 is also signalled by TINT0SRC . 0_B No Timer Service Request 0 1_B Timer Service Request 0 occurred This flag is cleared by writing a 1.
TI1	9	rwh	**Timer Service Request 1** This flag is set whenever the conditions programmed in the Timer Service Request 1 Configuration Register T1C are met. A Timer Service Request 1 is also signalled by TINT1SRC. 0_B No Timer Service Request 1 1_B Timer Service Request 1 occurred This flag is cleared by writing a 1.
TIBC	10	rwh	**Transfer Input Buffer Completed** This flag is set whenever a transfer from Input Buffer to the Message RAM has completed and bit IBCR.IBSYS in the Input Buffer Command Request register has been reset by the Message Handler. 0_B No transfer completed 1_B Transfer between Input Buffer and Message RAM completed This flag is cleared by writing a 1.
TOBC	11	rwh	**Transfer Output Buffer Completed** This flag is set whenever a transfer from Message RAM to the Output Buffer has completed and bit OBCR.OBSYS in the Output Buffer Command Request register has been reset by the Message Handler. 0_B No transfer completed 1_B Transfer between Message RAM and the Output Buffer completed This flag is cleared by writing a 1.
SWE	12	rwh	**Stop Watch Event** This flag is set after a stop watch activation when the current cycle counter and Macrotick value are stored in the Stop Watch Register 1 (STPW1). 0_B No Stop Watch Event 1_B Stop Watch Event occurred This flag is cleared by writing a 1.

Field	Bits	Type	Description
SUCS	13	rwh	**Startup Completed Successfully** This flag is set whenever a startup completed successfully and the Communication Controller entered “NORMAL_ACTIVE” state. 0_B No startup completed successfully 1_B Startup completed successfully This flag is cleared by writing a 1.
MBSI	14	rwh	**Message Buffer Status Service Request** This flag is set by the Communication Controller when the Message Buffer status MBS has changed and if bit RDHS1.MBI of that Message Buffer is set (see **Table 23-24**). 0_B No Message Buffer status change of Message Buffer with RDHS1.MBI= 1 has changed 1_B Message Buffer status of at least one Message Buffer with RDHS1.MBI= 1 has changed This flag is cleared by writing a 1.
SDS	15	rwh	**Start of Dynamic Segment** This flag is set by the Communication Controller when the dynamic segment starts. 0_B Dynamic segment not yet started 1_B Dynamic segment started
WUPA	16	rwh	**Wakeup Pattern Channel A** This flag is set by the Communication Controller when a wakeup pattern was received on channel A. Only set when the Communication Controller is in “WAKEUP”, “READY”, or “STARTUP” state, or when in Monitor mode. 0_B No wake-up pattern received on channel A 1_B Wake-up pattern received on channel A This flag is cleared by writing a 1.
MTSA	17	rwh	**MTS Received on Channel A** (vSS!ValidMTSA) Media Access Test symbol received on channel A during the proceeding symbol window. Updated by the Communication Controller for each channel at the end of the symbol window. 0_B No MTS symbol received on channel A 1_B MTS symbol received on channel A This flag is cleared by writing a 1.

Field	Bits	Type	Description
WUPB	24	rwh	**Wakeup Pattern Channel B** This flag is set by the Communication Controller when a wakeup pattern was received on channel B. Only set when the Communication Controller is in “WAKEUP”, “READY”, or “STARTUP” state, or when in Monitor mode. 0_B No wake-up pattern received on channel B 1_B Wake-up pattern received on channel B This flag is cleared by writing a 1.
MTSB	25	rwh	**MTS Received on Channel B** (vSS!ValidMTSB) Media Access Test symbol received on channel B during the proceeding symbol window. Updated by the Communication Controller for each channel at the end of the symbol window. 0_B No MTS symbol received on channel B 1_B MTS symbol received on channel B This flag is cleared by writing a 1.
0	[23:18], [31:26]	r	**Reserved** Returns 0 if read; should be written with 0.

Error Service Request Line Select (EILS)

The Error Service Request Line Select register assigns a service request generated by a specific error service request flag from register EIR to one of the two module service request lines INT0SRC or INT1SRC:

0 = Interrupt assigned to interrupt line (INT0SRC)

1 = Interrupt assigned to interrupt line (INT1SRC)

EILS

Error Service Request Line Select (0028_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					TABBL	LTVBL	EDBL	0					TABAL	LTVAL	EDAL
r					rw	rw	rw	r					rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				MHFL	IOBAL	IIBAL	EFAL	RFOL	EERRL	CCLL	CCFL	SFOL	SFBML	CNAL	PEMCL
r				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PEMCL	0	rw	**POC Error Mode Changed Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
CNAL	1	rw	**Command Not Accepted Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
SFBML	2	rw	**SYNC Frames Below Minimum Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC

Field	Bits	Type	Description
SFOL	3	rw	**SYNC Frame Overflow Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
CCFL	4	rw	**Clock Correction Failure Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
CCLL	5	rw	**CHI Command Locked Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
EERRL	6	rw	**ECC Error Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
RFOL	7	rw	**Receive FIFO Overrun Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
EFAL	8	rw	**Empty FIFO Access Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
IIBAL	9	rw	**Illegal Input Buffer Access Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC

Field	Bits	Type	Description
IOBAL	10	rw	**Illegal Output Buffer Access Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
MHFL	11	rw	**Message Handler Constrains Flag Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
EDAL	16	rw	**Error Detected on Channel A Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
LTVAL	17	rw	**Latest Transmit Violation Channel A Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
TABAL	18	rw	**Transmission Across Boundary Channel A Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
EDBL	24	rw	**Error Detected on Channel B Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
LTVBL	25	rw	**Latest Transmit Violation Channel B Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC

Field	Bits	Type	Description
TABBL	26	rw	**Transmission Across Boundary Channel A Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
0	[15:12], [23:19], [31:27]	r	**Reserved** Returns 0 if read; should be written with 0.

Status Service Request Line Select (SILS)

The Status Service Request Line Select register assign an service request generated by a specific status service request flag from register SIR to one of the two module service request lines INT0SRC or INT1SRC:

0 = Interrupt assigned to interrupt line INT0SRC

1 = Interrupt assigned to interrupt line INT1SRC

SILS

Status Service Request Line Select (002C$_H$) **Reset Value: 0303 FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						MTSBL	WUPBL	0						MTSAL	WUPAL
r						rw	rw	r						rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDSL	MBSIL	SUCSL	SWEL	TOBCL	TIBCL	TI1L	TI0L	NMVCL	RFCLL	RFNEL	RXIL	TXIL	CYCSL	CASL	WSTL
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
WSTL	0	rw	**Wakeup Status Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
CASL	1	rw	**Collision Avoidance Symbol Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
CYCSL	2	rw	**Cycle Start Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC

Field	Bits	Type	Description
TXIL	3	rw	**Transmit Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
RXIL	4	rw	**Receive Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
RFNEL	5	rw	**Receive FIFO Not Empty Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
RFCLL	6	rw	**Receive FIFO Critical Level Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
NMVCL	7	rw	**Network Management Vector Changed Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
TI0L	8	rw	**Timer Service Request 0 Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
TI1L	9	rw	**Timer Service Request 1 Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC

Field	Bits	Type	Description
TIBCL	10	rw	**Transfer Input Buffer Completed Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
TOBCL	11	rw	**Transfer Output Buffer Completed Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
SWEL	12	rw	**Stop Watch Event Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
SUCSL	13	rw	**Startup Completed Successfully Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
MBSIL	14	rw	**Message Buffer Status Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
SDSL	15	rw	**Start of Dynamic Segment Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
WUPAL	16	rw	**Wakeup Pattern Channel A Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC

Field	Bits	Type	Description
MTSAL	17	rw	**Media Access Test Symbol Channel A Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
WUPBL	24	rw	**Wakeup Pattern Channel B Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
MTSBL	25	rw	**Media Access Test Symbol Channel B Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
0	[23:18], [31:26]	r	**Reserved** Returns 0 if read; should be written with 0.

Error Service Request Enable Set (EIES)

The settings in the Error Service Request Enable register determine which status changes in the Error Service Request Register will result in an service request. The enable bits are set by writing to EIES and reset by writing to EIER. Writing a 1 sets the specific enable bit, a 0 has no effect.

EIES

Error Service Request Enable Set (0030_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					TABBE	LTVBE	EDBE	0					TABAE	LTVAE	EDAE
r					rwh	rwh	rwh	r					rwh	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				MHFE	IOBAE	IIBAE	EFAE	RFOE	EERRE	CCLE	CCFE	SFOE	SFBME	CNAE	PEMCE
r				rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
PEMCE	0	rwh	**POC Error Mode Changed Service Request Enable** 0_B Read: Protocol Error Mode Changed Service Request disabled Write: Unchanged 1_B Read: Protocol Error Mode Changed Service Request enabled Write: Enable Protocol Error Mode Changed Service Request
CNAE	1	rwh	**Command Not Accepted Service Request Enable** 0_B Read: Command Not Valid Service Request disabled Write: Unchanged 1_B Read: Command Not Valid Service Request enabled Write: Enable Command Not Valid Service Request
SFBME	2	rwh	**SYNC Frames Below Minimum Service Request Enable** 0_B Read: SYNC Frames Below Minimum Service Request disabled Write: Unchanged 1_B Read: SYNC Frames Below Minimum Service Request enabled Write: Enable SYNC Frames Below Minimum Service Request

Field	Bits	Type	Description
SFOE	3	rwh	**SYNC Frame Overflow Service Request Enable** 0_B Read: SYNC Frame Overflow Service Request disabled Write: Unchanged 1_B Read: SYNC Frame Overflow Service Request enabled Write: Enable Protocol Error Mode Changed Service Request
CCFE	4	rwh	**Clock Correction Failure Service Request Enable** 0_B Read: Clock Correction Failure Service Request disabled Write: Unchanged 1_B Read: Clock Correction Failure Service Request enabled Write: Enable Clock Correction Failure Service Request
CCLE	5	rwh	**CHI Command Locked Service Request Enable** 0_B Read: CHI Command Locked Service Request disabled Write: Unchanged 1_B Read: CHI Command Locked Service Request enabled Write: Enable CHI Command Locked Service Request
EERRE	6	rwh	**ECC Error Service Request Enable** 0_B Read: ECC Error Service Request disabled Write: Unchanged 1_B Read: ECC Error Service Request enabled Enable ECC Error Service Request Write: Unchanged
RFOE	7	rwh	**Receive FIFO Overrun Service Request Enable** 0_B Read: Receive FIFO Overrun Service Request disabled Write: Unchanged 1_B Read: Receive FIFO Overrun Service Request enabled Write: Enable Receive FIFO Overrun Service Request
EFAE	8	rwh	**Empty FIFO Access Service Request Enable** 0_B Read: Empty FIFO Access Service Request disabled Write: Unchanged 1_B Read: Empty FIFO Access Service Request enabled Write: Enable Empty FIFO Access Service Request

Field	Bits	Type	Description
IIBAE	9	rwh	**Illegal Input Buffer Access Service Request Enable** 0_B Read: Illegal Input Buffer Access Service Request disabled Write: Unchanged 1_B Read: Illegal Input Buffer Access Service Request enabled Write: Enable Illegal Input Buffer Access Service Request
IOBAE	10	rwh	**Illegal Output Buffer Access Service Request Enable** 0_B Read: Illegal Output Buffer Access Service Request disabled Write: Unchanged 1_B Read: Illegal Output Buffer Access Service Request enabled Write: Enable Illegal Output Buffer Access Service Request
MHFE	11	rwh	**Message Handler Constraints Flag Service Request Enable** 0_B Read: Message Handler Constraints Flag Service Request disabled Write: Unchanged 1_B Read: Message Handler Constraints Flag Service Request enabled Write: Enable Message Handler Constraints Flag Service Request
EDAE	16	rwh	**Error Detected on Channel A Service Request Enable** 0_B Read: Error Detected on Channel A Service Request disabled Write: Unchanged 1_B Read: Error Detected on Channel A Service Request enabled Write: Enable Error Detected on Channel A Service Request

Field	Bits	Type	Description
LTVAE	17	rwh	**Latest Transmit Violation Channel A Service Request Enable** 0_B Read: Latest Transmit Violation Channel A Service Request disabled Write: Unchanged 1_B Read: Latest Transmit Violation Channel A Service Request enabled Write: Enable Latest Transmit Violation Channel A Service Request
TABAE	18	rwh	**Transmission Across Boundary Channel A Service Request Enable** 0_B Read: Transmission Across Boundary Channel A Service Request disabled Write: Unchanged 1_B Read: Transmission Across Boundary Channel A Service Request enabled Write: Enable Transmission Across Boundary Channel A Service Request
EDBE	24	rwh	**Error Detected on Channel B Service Request Enable** 0_B Read: Error Detected on Channel B Service Request disabled Write: Unchanged 1_B Read: Error Detected on Channel B Service Request enabled Write: Enable Error Detected on Channel B Service Request
LTVBE	25	rwh	**Latest Transmit Violation Channel B Service Request Enable** 0_B Read: Latest Transmit Violation Channel B Service Request disabled Write: Unchanged 1_B Read: Latest Transmit Violation Channel B Service Request enabled Write: Enable Latest Transmit Violation Channel B Service Request

Field	Bits	Type	Description
TABBE	26	rwh	**Transmission Across Boundary Channel B Service Request Enable** 0_B Read: Transmission Across Boundary Channel B Service Request disabled Write: Enable Transmission Across Boundary Channel B Service Request 1_B Read: Transmission Across Boundary Channel B Service Request enabled Write: Enable Transmission Across Boundary Channel B Service Request
0	[15:12], [23:19], [31:27]	r	**Reserved** Returns 0 if read; should be written with 0.

Error Service Request Enable Reset (EIER)

The settings in the Error Service Request Enable register determine which status changes in the Error Service Request Register will result in a service request. The enable bits are set by writing to EIES and reset by writing to EIER. Writing a 1 resets the specific enable bit, a 0 has no effect.

EIER

Error Service Request Enable Reset (0034$_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					TABBE	LTVBE	EDBE	0					TABAE	LTVAE	EDAE
r					rwh	rwh	rwh	r					rwh	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				MHFE	IOBAE	IIBAE	EFAE	RFOE	EERRE	CCLE	CCFE	SFOE	SFBME	CNAE	PEMCE
r				rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
PEMCE	0	rwh	**POC Error Mode Changed Service Request Enable** 0_B Read: Protocol Error Mode Changed Service Request disabled Write: Unchanged 1_B Read: Protocol Error Mode Changed Service Request enabled Write: Disable Protocol Error Mode Changed Service Request
CNAE	1	rwh	**Command Not Accepted Service Request Enable** 0_B Read: Command Not Accepted Service Request disabled Write: Unchanged 1_B Read: Command Not Accepted Service Request enabled Write: Disable Command Not Accepted Service Request

Field	Bits	Type	Description
SFBME	2	rwh	**SYNC Frames Below Minimum Service Request Enable** 0_B Read: SYNC Frames Below Minimum Service Request disabled Write: Unchanged 1_B Read: SYNC Frames Below Minimum Service Request enabled Write: Disable SYNC Frames Below Minimum Service Request
SFOE	3	rwh	**SYNC Frame Overflow Service Request Enable** 0_B Read: SYNC Frame Overflow Service Request disabled Write: Unchanged 1_B Read: SYNC Frame Overflow Service Request enabled Write: Disable Protocol Error Mode Changed Service Request
CCFE	4	rwh	**Clock Correction Failure Service Request Enable** 0_B Read: Clock Correction Failure Service Request disabled Write: Unchanged 1_B Read: Clock Correction Failure Service Request enabled Write: Disable Clock Correction Failure Service Request
CCLE	5	rwh	**CHI Command Locked Service Request Enable** 0_B Read: CHI Command Locked Service Request disabled Write: Unchanged 1_B Read: CHI Command Locked Service Request enabled Write: Disable CHI Command Locked Service Request
EERRE	6	rwh	**ECC Error Service Request Enable** 0_B Read: ECC Error Service Request disabled Write: Unchanged 1_B ERead: CC Error Service Request enabled Write: Disable ECC Error Service Request
RFOE	7	rwh	**Receive FIFO Overrun Service Request Enable** 0_B Read: Receive FIFO Overrun Service Request disabled Write: Unchanged 1_B Read: Receive FIFO Overrun Service Request enabled Write: Disable Receive FIFO Overrun Service Request

Field	Bits	Type	Description
EFAE	8	rwh	**Empty FIFO Access Service Request Enable** 0_B Read: Empty FIFO Access Service Request disabled Write: Unchanged 1_B Read: Empty FIFO Access Service Request enabled Write: Disable Empty FIFO Access Service Request
IIBAE	9	rwh	**Illegal Input Buffer Access Service Request Enable** 0_B Read: Illegal Input Buffer Access Service Request disabled Write: Unchanged 1_B Read: Illegal Input Buffer Access Service Request enabled Write: Disable Illegal Input Buffer Access Service Request
IOBAE	10	rwh	**Illegal Output Buffer Access Service Request Enable** 0_B Read: Illegal Output Buffer Access Service Request disabled Write: Unchanged 1_B Read: Illegal Output Buffer Access Service Request enabled Write: Disable Illegal Output Buffer Access Service Request
MHFE	11	rwh	**Message Handler Constraints Flag Service Request Enable** 0_B Read: Message Handler Constraints Flag Service Request disabled Write: Unchanged 1_B Read: Message Handler Constraints Flag Service Request enabled Write: Disable Message Handler Constraints Flag Service Request
EDAE	16	rwh	**Error Detected on Channel A Service Request Enable** 0_B Read: Error Detected on Channel A Service Request disabled Write: Unchanged 1_B Read: Error Detected on Channel A Service Request enabled Write: Disable Error Detected on Channel A Service Request

Field	Bits	Type	Description
LTVAE	17	rwh	**Latest Transmit Violation Channel A Service Request Enable** 0_B Read: Latest Transmit Violation Channel A Service Request disabled Write: Unchanged 1_B Read: Latest Transmit Violation Channel A Service Request enabled Write: Disable Latest Transmit Violation Channel A Service Request
TABAE	18	rwh	**Transmission Across Boundary Channel A Service Request Enable** 0_B Read: Transmission Across Boundary Channel A Service Request disabled Write: Unchanged 1_B Read: Transmission Across Boundary Channel A Service Request enabled Write: Enable Transmission Across Boundary Channel A Service Request
EDBE	24	rwh	**Error Detected on Channel B Service Request Enable** 0_B Read: Error Detected on Channel B Service Request disabled Write: Unchange 1_B Read: Error Detected on Channel B Service Request enabled Write: Disable Error Detected on Channel B Service Request
LTVBE	25	rwh	**Latest Transmit Violation Channel B Service Request Enable** 0_B Read: Latest Transmit Violation Channel B Service Request disabled Write: Unchanged 1_B Read: Latest Transmit Violation Channel B Service Request enabled Write: Disable Latest Transmit Violation Channel B Service Request

Field	Bits	Type	Description
TABBE	26	rwh	**Transmission Across Boundary Channel B Service Request Enable** 0_B Read: Transmission Across Boundary Channel B Service Request disabled Write: Unchanged 1_B Read: Transmission Across Boundary Channel B Service Request enabled Write: Disable Transmission Across Boundary Channel B Service Request
0	[15:12], [23:19], [31:27]	r	**Reserved** Returns 0 if read; should be written with 0.

Status Service Request Enable Set (SIES)

The settings in the Status Service Request Enable Set register determine which status changes in the Status Service Request Register will result in a service request. The enable bits are set by writing to SIES and reset by writing to SIER. Writing a 1 sets the specific enable bit, a 0 has no effect.

SIES

Status Service Request Enable Set (0038_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						MTSBE	WUPBE	0						MTSAE	WUPAE
r						rwh	rwh	r						rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDSE	MBSIE	SUCSE	SWEE	TOBCE	TIBCE	TI1E	TI0E	NMVCE	RFCLE	RFNEE	RXIE	TXIE	CYCSE	CASE	WSTE
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
WSTE	0	rwh	**Wakeup Status Service Request Enable** 0_B Read: Wake-up Status Service Request disabled Write: Unchanged 1_B Read:Wake-up Status Service Request enabled Write: Enable Wakeup Status Service Request
CASE	1	rwh	**Collision Avoidance Symbol Service Request Enable** 0_B Read: Collision Avoidance Symbol Service Request disabled Write: Unchanged 1_B Read: Collision Avoidance Symbol Service Request enabled Write: Enable Collision Avoidance Symbol Service Request
CYCSE	2	rwh	**Cycle Start Service Request Enable** 0_B Read: Cycle Start Service Request disabled Write: Unchanged 1_B Read: Cycle Start Service Request enabled Write: Enable Cycle Start Service Request

Field	Bits	Type	Description
TXIE	3	rwh	**Transmit Service Request Enable** 0_B Read: Transmit Service Request disabled Write: Unchanged 1_B Transmit Service Request enabled Write: Enable Transmit Service Request
RXIE	4	rwh	**Receive Service Request Enable** 0_B Read: Receive Service Request disabled Write: Unchanged 1_B Read: Receive Service Request enabled Write: Enable Receive Service Request
RFNEE	5	rwh	**Receive FIFO Not Empty Service Request Enable** 0_B Read: Receive FIFO Not Empty Service Request disabled Write: Unchanged 1_B Read: Receive FIFO Not Empty Service Request enabled Write: Enable Receive FIFO Not Empty Service Request
RFCLE	6	rwh	**Receive FIFO Critical Level Service Request Enable** 0_B Read: Receive FIFO Critical Level Service Request disabled Write: Unchanged 1_B Read: Receive FIFO Critical Level Service Request enabled Write: Enable Receive FIFO Critical Level Service Request
NMVCE	7	rwh	**Network Management Vector Changed Service Request Enable** 0_B Read: Network Management Vector Changed Service Request disabled Write: Unchanged 1_B Read: Network Management Vector Changed Service Request enabled Write: Enable Network Management Vector Changed Service Request

Field	Bits	Type	Description
TI0E	8	rwh	**Timer Service Request 0 Enable** 0_B Read: Timer Service Request 0disabled Write: Unchanged 1_B Read: Timer Service Request 0 enabled Write: Enable Timer Service Request 0
TI1E	9	rwh	**Timer Service Request 1 Enable** 0_B Read: Timer Service Request 1disabled Write: Unchanged 1_B Read: Timer Service Request 1 enabled Write: Enable Timer Service Request 1
TIBCE	10	rwh	**Transfer Input Buffer Completed Service Request Enable** 0_B Read: Wakeup Status Service Request disabled Write: Unchanged 1_B Read: Wakeup Status Service Request enabled Write: Enable Wakeup Status Service Request
TOBCE	11	rwh	**Transfer Output Buffer Completed Service Request Enable** 0_B Read: Transfer Input Buffer Completed Service Request disabled Write: Unchanged 1_B Read: Transfer Input Buffer Completed Service Request enabled Write: Enable Transfer Input Buffer Completed Service Request
SWEE	12	rwh	**Stop Watch Event Service Request Enable** 0_B Read: Stop Watch Event Service Request disabled Write: Unchanged 1_B Read: Stop Watch Event Service Request enabled Write: Enable Stop Watch Event Service Request
SUCSE	13	rwh	**Startup Completed Successfully Service Request Enable** 0_B Read: Startup Completed Successfully Service Request disabled Write: Unchanged 1_B Read: Startup Completed Successfully Service Request enabled Write: Enable Startup Completed Successfully Service Request

Field	Bits	Type	Description
MBSIE	14	rwh	**Message Buffer Status Service Request Enable** 0_B Read: Message Buffer Status Service Request disabled Write: Unchanged 1_B Read: Message Buffer Status Service Request enabled Write: Enable Message Buffer Status Service Request
SDSE	15	rwh	**Start of Dynamic Segment Service Request Enable** 0_B Read: Start of Dynamic Service Request disabled Write: Unchanged 1_B Read: Start of Dynamic Service Request enabled Write: Enable Start of Dynamic Service Request
WUPAE	16	rwh	**Wakeup Pattern Channel A Service Request Enable** 0_B Read: Wakeup Pattern Channel A Service Request disabled Write: Unchanged 1_B Read: Wakeup Pattern Channel A Service Request enabled Write: Enable Wakeup Pattern Channel A Service Request
MTSAE	17	rwh	**Media Access Test Symbol Channel A Service Request Enable** 0_B Read: Media Access Test Symbol Channel A Service Request disabled Write: Unchanged 1_B Read: Media Access Test Symbol Channel A Service Request enabled Write: Enable Media Access Test Symbol Channel A Service Request
WUPBE	24	rwh	**Wakeup Pattern Channel B Service Request Enable** 0_B Read: Wakeup Pattern Channel B Service Request disabled Write: Unchanged 1_B Read: Wakeup Pattern Channel B Service Request enabled Write: Enable Wakeup Pattern Channel B Service Request

Field	Bits	Type	Description
MTSBE	25	rwh	**Media Access Test Symbol Channel B Service Request Enable** 0_B Read: Media Access Test Symbol Channel B Service Request disabled Write: Unchanged 1_B Read: Media Access Test Symbol Channel B Service Request enabled Write: Enable Media Access Test Symbol Channel B Service Request
0	[23:18], [31:26]	r	**Reserved** Returns 0 if read; should be written with 0.

Status Service Request Enable Reset (SIER)

The settings in the Status Service Request Enable Reset register determine which status changes in the Status Service Request Register will result in a service request. The enable bits are set by writing to SIES and reset by writing to SIER. Writing a 1 resets the specific enable bit, a 0 has no effect.

SIER

Status Service Request Enable Reset(003C$_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						MTSBE	WUPBE	0						MTSAE	WUPAE
r						rwh	rwh	r						rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDSE	MBSIE	SUCSE	SWEE	TOBCE	TIBCE	TI1E	TI0E	NMVCE	RFCLE	RFNEE	RXIE	TXIE	CYCSE	CASE	WSTE
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
WSTE	0	rwh	**Wakeup Status Service Request Enable** 0_B Read: Wakeup Status Service Request disabled Write: Unchanged 1_B Read: Wakeup Status Service Request enabled Write: Disable Wakeup Status Service Request
CASE	1	rwh	**Collision Avoidance Symbol Service Request Enable** 0_B Read: Collision Avoidance Symbol Service Request disabled Write: Unchanged 1_B Read: Collision Avoidance Symbol Service Request enabled Write: Disable Collision Avoidance Symbol Service Request
CYCSE	2	rwh	**Cycle Start Service Request Enable** 0_B Read: Cycle Start Service Request disabled Write: Unchanged 1_B Read: Cycle Start Service Request enabled Write: Disable Cycle Start Service Request

Field	Bits	Type	Description
TXIE	3	rwh	**Transmit Service Request Enable** 0_B Read: Transmit Service Request disabled Write: Unchanged 1_B Read: Transmit Service Request enabled Write: Disable Transmit Service Request
RXIE	4	rwh	**Receive Service Request Enable** 0_B Read: Receive Service Request disabled Write: Unchanged 1_B Read: Receive Service Request enabled Write: Disable Receive Service Request
RFNEE	5	rwh	**Receive FIFO Not Empty Service Request Enable** 0_B Read: Receive FIFO Not Empty Service Request disabled Write: Unchanged 1_B Read: Receive FIFO Not Empty Service Request enabled Write: Disable Receive FIFO Not Empty Service Request
RFCLE	6	rwh	**Receive FIFO Critical Level Service Request Enable** 0_B Read: Service Request disabled Write: Unchanged 1_B Read: Receive FIFO Critical Level Service Request enabled Write: Disable Receive FIFO Critical Level Service Request
NMVCE	7	rwh	**Network Management Vector Changed Service Request Enable** 0_B Read: Network Management Vector Changed Service Request disabled Write: Unchanged 1_B Read: Network Management Vector Changed Service Request enabled Write: Disable Network Management Vector Changed Service Request
TI0E	8	rwh	**Timer Service Request 0 Enable** 0_B Read: Timer Service Request 0 disabled Write: Unchanged 1_B Read: Timer Service Request 0 enabled Write: Disable Service Request 0

Field	Bits	Type	Description
TI1E	9	rwh	**Timer Service Request 1 Enable** 0_B Read: Timer Service Request 1 disabled Write: Unchanged 1_B Read: Timer Service Request 1 enabled Write: Disable Timer Service Request 1
TIBCE	10	rwh	**Transfer Input Buffer Completed Service Request Enable** 0_B Read: Wakeup Status Service Request disabled Write: Unchanged 1_B Read: Wakeup Status Service Request enabled Write: Disable Wakeup Status Service Request
TOBCE	11	rwh	**Transfer Output Buffer Completed Service Request Enable** 0_B Read: Transfer Input Buffer Completed Service Request disabled Write: Unchanged 1_B Read: Transfer Input Buffer Completed Service Request enabled Write: Disable Transfer Input Buffer Completed Service Request
SWEE	12	rwh	**Stop Watch Event Service Request Enable** 0_B Read: Stop Watch Event Service Request disabled Write: Unchanged 1_B Read: Stop Watch Event Service Request enabled Write: Disable Stop Watch Event Service Request
SUCSE	13	rwh	**Startup Completed Successfully Service Request Enable** 0_B Read: Startup Completed Successfully Service Request disabled Write: Unchanged 1_B Read: Startup Completed Successfully Service Request enabled Write: Disable Startup Completed Successfully Service Request
MBSIE	14	rwh	**Message Buffer Status Service Request Enable** 0_B Read: Message Buffer Status Service Request disabled Write: Unchanged 1_B Read: Message Buffer Status Service Request enabled Write: Disable Message Buffer Status Service Request

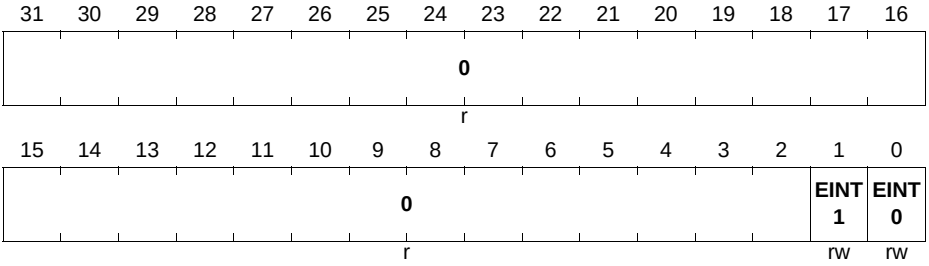
Field	Bits	Type	Description
SDSE	15	rwh	**Start of Dynamic Segment Service Request Enable** 0_B Read: Start of Dynamic Service Request disabled Write: Unchanged 1_B Read: Start of Dynamic Service Request enabled Write: Disable Start of Dynamic Service Reques
WUPAE	16	rwh	**Wakeup Pattern Channel A Service Request Enable** 0_B Read: Wakeup Pattern Channel A Service Request disabled Write: Unchanged 1_B Read: Wakeup Pattern Channel A Service Request enabled Write: Disable Wakeup Pattern Channel A Service Request
MTSAE	17	rwh	**Media Access Test Symbol Channel A Service Request Enable** 0_B Read: Media Access Test Symbol Channel A Service Request disabled Write: Unchanged 1_B Read: Media Access Test Symbol Channel A Service Request enabled Write: Disable Media Access Test Symbol Channel A Service Request
WUPBE	24	rwh	**Wakeup Pattern Channel B Service Request Enable** 0_B Read: Wakeup Pattern Channel B Service Request disabled Write: Unchanged 1_B Read: Wakeup Pattern Channel B Service Request enabled Write: Disable Wakeup Pattern Channel B Service Request
MTSBE	25	rwh	**Media Access Test Symbol Channel B Service Request Enable** 0_B Read: Media Access Test Symbol Channel B Service Request disabled Write: Unchanged 1_B Read: Media Access Test Symbol Channel B Service Request enabled Write: Disable Media Access Test Symbol Channel B Service Request

Field	Bits	Type	Description
0	[23:18], [31:26]	r	**Reserved** Returns 0 if read; should be written with 0.

Service Request Line Enable (ILE)

Each of the two service request lines to the Host INT0SRC, INT1SRC can be enabled / disabled separately by programming bit EINT0 and EINT1.

ILE
Service Request Line Enable **(0040_H)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
EINT0	0	rw	**Enable Service Request Line 0** (INT0SRC) 0_B Service Request line disabled 1_B Service Request line enabled
EINT1	1	rw	**Enable Service Request Line 1** (INT1SRC) 0_B Service Request line disabled 1_B Service Request line enabled
0	[31:2]	r	**Reserved** Returns 0 if read; should be written with 0.

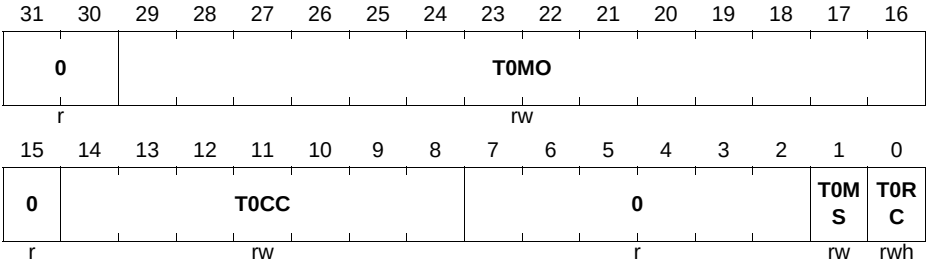
Timer 0 Configuration (T0C)

Absolute timer. Specifies in terms of cycle count and Macrotick the point in time when the timer 0 service request occurs. When the timer 0 service request is asserted, output signal TINT0SR is set to 1 for the duration of one Macrotick and SIR.TI0 is set to 1.

Timer 0 can be activated as long as the POC is either in “NORMAL_ACTIVE” state or in “NORMAL_PASSIVE” state. Timer 0 is deactivated when leaving “NORMAL_ACTIVE” state or “NORMAL_PASSIVE” state except for transitions between the two states.

Before reconfiguration of the timer, the timer has to be halted first by writing 0 to bit T0RC.

T0C
Timer 0 Configuration **(0044_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
T0RC	0	rwh	**Timer 0 Run Control** 0_B Timer 0 halted 1_B Timer 0 running
T0MS	1	rw	**Timer 0 Mode Select** 0_B Single-shot mode 1_B Continuous mode
T0CC	[14:8]	rw	**Timer 0 Cycle Code** The 7-bit timer 0 cycle code determines the cycle set used for generation of the timer 0 service request. For details about the configuration of the cycle code see **“Cycle Counter Filtering” on Page 23-215**.
T0MO	[29:16]	rw	**Timer 0 Macrotick Offset** Configures the Macrotick offset from the beginning of the cycle where the service request is to occur. The Timer 0 Service Request occurs at this offset for each cycle of the cycle set.

Field	Bits	Type	Description
0	[7:2], 15, [31:30]	r	**Reserved** Returns 0 if read; should be written with 0.

Note: The configuration of timer 0 is compared against the Macrotick counter value, there is no separate counter for timer 0. In case the Communication Controller leaves “NORMAL_ACTIVE” or “NORMAL_PASSIVE” state, or if timer 0 is halted by Host command, output signal TINT0SR *is reset to 0 immediately.*

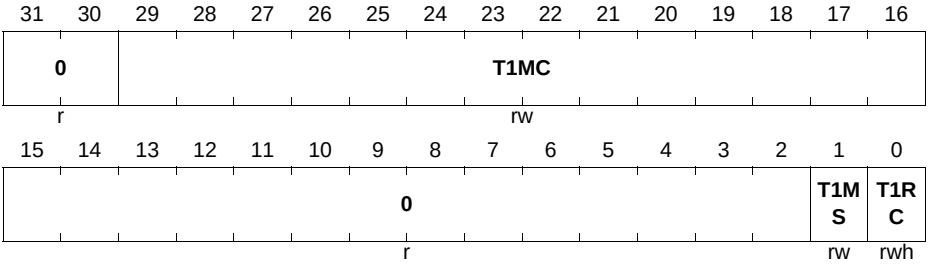
Timer 1 Configuration (T1C)

Relative timer. After the specified number of Macroticks has expired, the timer 1 service request is asserted, output signal TINT1SR is set to 1 for the duration of one Macrotick and SIR.TI1 is set to 1.

Timer 1 can be activated as long as the POC is either in “NORMAL_ACTIVE” state or in “NORMAL_PASSIVE” state. Timer 1 is deactivated when leaving “NORMAL_ACTIVE” state or “NORMAL_PASSIVE” state except for transitions between the two states.

Before reconfiguration of the timer, the timer has to be halted first by resetting bit T1RC to 0.

T1C
Timer 1 Configuration **(0048_H)** **Reset Value: 0002 0000$_H$**



Field	Bits	Type	Description
T1RC	0	rwh	**Timer 1 Run Control** 0_B Timer 1 halted 1_B Timer 1 running
T1MS	1	rw	**Timer 1 Mode Select** 0_B Single-shot mode 1_B Continuous mode
T1MC	[29:16]	rw	**Timer 1 Macrotick Count** When the configured Macrotick count is reached the timer 1 service request is generated. Valid values are: 2_H …$3FFF_H$ Macroticks in continuous mode 1_H …$3FFF_H$ Macroticks in single-shot mode
0	[15:2], [31:30]	r	**Reserved** Returns 0 if read; should be written with 0.

Note: In case the Communication Controller leaves "NORMAL_ACTIVE" or "NORMAL_PASSIVE" state, or if timer 1 is halted by Host command, output signal TINT1SR *is reset to 0 immediately.*

Stop Watch Register 1 (STPW1)

The stop watch is activated by a rising or falling edge on signal STPW, by a service request 0 or 1 event (rising edge on signal INT0SR or INT1SR) or by the Host by writing bit STPW1.SSWT to 1. With the Macrotick counter increment following next to the stop watch activation the actual cycle counter and Macrotick value are captured in the Stop Watch Register 1 STPW1 while the slot counter values for channel A and B are captured in the Stop Watch Register 2 STPW2.

STPW1
Stop Watch Register 1 **(004C$_H$)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								SMTV							
r								rh							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		SCCV						0	EINT1	EINT0	EETP	SSWT	EDGE	SWMS	ESWT
r		rh						r	rw	rw	rw	rwh	rw	rw	rwh

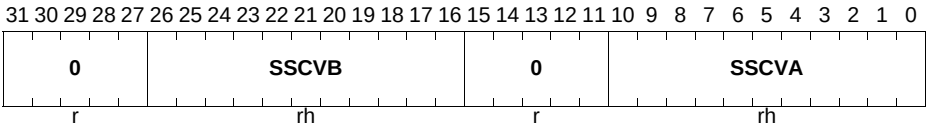
Field	Bits	Type	Description
ESWT	0	rwh	**Enable Stop Watch Trigger** If enabled an edge on input STPW or a service request 0 or 1 event (rising edge on signal INT0SR or INT1SR) activates the stop watch. In single-shot mode this bit is reset to 0 after the actual cycle counter and Macrotick value are stored in the Stop Watch register. 0_B Stop watch trigger disabled 1_B Stop watch trigger enabled
SWMS	1	rw	**Stop Watch Mode Select** It is not possible to change the Stop Watch Mode during enabled stop watch trigger (STPW1.ESWT) 0_B Single-shot mode 1_B Continuous mode
EDGE	2	rw	**Stop Watch Trigger Edge Select** 0_B Falling Edge 1_B Rising Edge

Field	Bits	Type	Description
SSWT	3	rwh	**Software Stop Watch Trigger** When the Host writes this bit to 1 the stop watch is activated. After the actual cycle counter and Macrotick value are stored in the Stop Watch register this bit is reset to 0. The bit is only writeable while ESWT = 0. 0_B Software trigger reset 1_B Stop watch activated by software trigger
EETP	4	rw	**Enable External Trigger Pin** Enables stop watch trigger event via signal STPW if ESWT = 1. 0_B Stop watch trigger via signal STPW disabled 1_B Edge on signal STPW triggers stop watch
EINT0	5	rw	**Enable Service Request 0 Trigger** Enables stop watch trigger by service request 0 event if ESWT = 1. 0_B Stop watch trigger by service request 0 disabled 1_B Service Request 0 event triggers stop watch
EINT1	6	rw	**Enable Service Request 1 Trigger** Enables stop watch trigger by service request 1 event if ESWT = 1. 0_B Stop watch trigger by service request 1 disabled 1_B Service Request 1 event triggers stop watch
SCCV	[13:8]	rh	**Stopped Cycle Counter Value** State of the cycle counter when the stop watch event occurred. Valid values are: $0…3F_H$ Valid Values
SMTV	[29:16]	rh	**Stopped Macrotick Value** State of the Macrotick counter when the stop watch event occurred. Valid values are: $0…3F_H$ Valid Values
0	7, [15:14], [31:30]	r	**Reserved** Returns 0 if read; should be written with 0.

Note: Bits ESWT and SSWT cannot be set to 1 simultaneously. In this case the write access is ignored, and both bits keep their previous values. Therefore either the external stop watch triggers or the software stop watch trigger may be used.

Stop Watch Register 2 (STPW2)

STPW2
Stop Watch Register 2 **(0050_H)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
SSCVA	[10:0]	rh	**Stop Watch Captured Slot Counter Value Channel A** State of the slot counter for channel A when the stop watch event occurred. Valid values are 0 to 2047 (0_H to $7FF_H$).
SSCVB	[26:16]	rh	**Stop Watch Captured Slot Counter Value Channel B** State of the slot counter for channel B when the stop watch event occurred. Valid values are 0 to 2047 (0_H to $7FF_H$).
0	[15:11], [31:27]	r	**Reserved** Returns 0 if read; should be written with 0.

23.5.2.4 Communication Controller Control Registers

This section describes the registers provided by the Communication Controller to allow the Host to control the operation of the Communication Controller. The FlexRay™ protocol specification requires the Host to write application configuration data in “CONFIG” state only. Please consider that the configuration registers are not locked for writing in “DEFAULT_CONFIG” state.

The configuration data is reset when “DEFAULT_CONFIG” state is entered from application reset. To change POC state from “DEFAULT_CONFIG” to “CONFIG” state the Host has to apply CHI command “CONFIG”. If the Host wants the Communication Controller to leave “CONFIG” state, the Host has to proceed as described on **“Lock Register (LCK)” on Page 23-30**.

SUC Configuration Register 1 (SUCC1)

SUCC1
SUC Configuration Register 1 (0080_H) Reset Value: $0C40\ 1000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				CCHB	CCHA	MTSB	MTSA	HCSE	TSM	WUCS	PTA				
r				rw	rw	rw	rw	rw	rw	rw	rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSA					0	TXSY	TXST	PBSY	0			CMD			
rw					r	rw	rw	rh	r			rwh			

Field	Bits	Type	Description
CMD	[3:0]	rwh	**CHI Command Vector** The host may write any CHI command at any time, but certain commands are only enabled in specific POC states. A disabled command will not be executed, the CHI command vector CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”, and flag EIR.CNA in the Error Service Request register will be set to 1. In case the previous CHI command has not yet completed, EIR.CCL is set to 1 together with EIR.CNA; the CHI command needs to be repeated. Except for HALT state, POC state change command applied while the Communication Controller is already in the requested POC state will be ignored. 0000_B COMMAND_NOT_ACCEPTED” 0001_B CONFIG 0010_B READY 0011_B WAKEUP 0100_B RUN 0101_B ALL_SLOTS 0110_B HALT 0111_B FREEZE 1000_B SEND_MTS 1001_B ALLOW_COLDSTART 1010_B RESET_STATUS_INDICATORS 1011_B MONITOR_MODE 1100_B CLEAR_RAMS 1101_B Reserved 1110_B Reserved 1111_B Reserved Reading SUCC1.CMD shows whether the last CHI command was accepted. CCSV.POCS monitors the actual POC state. The reserved CHI commands code hardware test functions.
PBSY	7	rh	**POC Busy** Signals that the POC is busy and cannot accept a command from the Host. SUCC1.CMD is locked against write accesses. Set to 1 after hard reset during initialization of internal RAM blocks. 0_B POC not busy, SUCC1.CMD writable 1_B POC is busy, SUCC1.CMD locked

Field	Bits	Type	Description
TXST	8	rw	**Transmit Startup Frame in Key Slot**[1) 2)] (pKeySlotUsedForStartup) Defines whether the key slot is used to transmit startup Frames. The bit can be modified in “DEFAULT_CONFIG” or “CONFIG” state only. 0_B No Startup Frame transmission in key slot, node is non-coldstarter 1_B Key slot used to transmit startup Frame, node is leading or following coldstarter
TXSY	9	rw	**Transmit SYNC Frame in Key Slot**[1) 2)] (pKeySlotUsedForSync) Defines whether the key slot is used to transmit SYNC Frames. The bit can be modified in “DEFAULT_CONFIG” or “CONFIG” state only. 0_B No SYNC Frame transmission in key slot, node is neither sync nor coldstart node 1_B Key slot used to transmit SYNC Frames, node is sync node
CSA	[15:11]	rw	**Cold Start Attempts**[1)](gColdStartAttempts) Configures the maximum number of attempts that a cold starting node is permitted to try to start up the network without receiving any valid response from another node. It can be modified in “DEFAULT_CONFIG” or “CONFIG” state only. Must be identical in all nodes of a cluster. Valid values are 2 to 31.
PTA	[20:16]	rw	**Passive to Active**[1)] (pAllowPassiveToActive) Defines the number of consecutive even / odd cycle pairs that must have valid clock correction terms before the Communication Controller is allowed to transit from “NORMAL_PASSIVE” to “NORMAL_ACTIVE” state. If set to 00000_B the Communication Controller is not allowed to transit from “NORMAL_PASSIVE” to “NORMAL_ACTIVE” state. It can be modified in “DEFAULT_CONFIG” or “CONFIG” state only. Valid values are 0 to 31 even / odd cycle pairs.

Field	Bits	Type	Description
WUCS	21	rw	**Wakeup Channel Select**[1)] (pWakeupChannel) With this bit the Host selects the channel on which the Communication Controller sends the Wakeup pattern. The Communication Controller ignores any attempt to change the status of this bit when not in “DEFAULT_CONFIG” or “CONFIG” state. 0_B Send wakeup pattern on channel A 1_B Send wakeup pattern on channel B
TSM	22	rw	**Transmission Slot Mode**[1)] (pSingleSlotEnabled) Selects the initial transmission slot mode. In SINGLE slot mode the Communication Controller may only transmit in the preconfigured key slot. The key slot ID is configured in the Header Section of Message Buffer 0 respectively Message Buffers 0 and 1 depending on bit MRC.SPLM. In case SUCC1.TSM = 1, Message Buffer 0 respectively Message Buffers 0,1 can be (re)configured in “DEFAULT_CONFIG” or “CONFIG” state only. In ALL slot mode the Communication Controller may transmit in all slots. The bit can be written in “DEFAULT_CONFIG” or “CONFIG” state only. The communication controller changes to ALL slot mode when the Host successfully applied the ALL_SLOTS command by writing SUCC1.CMD = 0101_B in POC states “NORMAL_ACTIVE” or “NORMAL_PASSIVE”. The actual slot mode is monitored by CCSV.SLM. 0_B ALL Slot Mode 1_B SINGLE Slot Mode (default after application reset)
HCSE	23	rw	**Halt due to Clock Sync Error**[1)] (pAllowHaltDueToClock) Controls the transition to “HALT” state due to a clock synchronization error. The bit can be modified in “DEFAULT_CONFIG” or “CONFIG” state only. 0_B Communication Controller will enter / remain in “NORMAL_PASSIVE” 1_B Communication Controller will enter “HALT” state
MTSA	24	rw	**Select Channel A for MTS Transmission**[1) 3)] The bit selects channel A for MTS symbol transmission. The flag is reset by default and may be modified only in “DEFAULT_CONFIG” or “CONFIG” state. 0_B Channel A disabled for MTS transmission 1_B Channel A selected for MTS transmission

Field	Bits	Type	Description
MTSB	25	rw	**Select Channel B for MTS Transmission[1) 3)]** The bit selects channel B for MTS symbol transmission. The flag is reset by default and may be modified only in “DEFAULT_CONFIG” or “CONFIG” state. 0_B Channel B disabled for MTS transmission 1_B Channel B selected for MTS transmission
CCHA	26	rw	**Connected to Channel A[1)]** (pChannels) Configures whether the node is connected to channel A. 0_B Not connected to channel A 1_B Node connected to channel A (default after application reset)
CCHB	27	rw	**Connected to Channel B[1)]** (pChannels) Configures whether the node is connected to channel B. 0_B Not connected to channel B 1_B Node connected to channel B (default after application reset)
0	[6:4], 10, [31:28]	r	**Reserved** Returns 0 if read; should be written with 0.

1) This bit can be updated in “DEFAULT_CONFIG” or “CONFIG” state only!

2) The protocol requires that both bits TXST and TXSY are set for coldstart nodes.

3) MTSA and MTSB may also be changed outside “DEFAULT_CONFIG” or “CONFIG” state when the write to SUCC1 register is directly preceded by the unlock sequence as described in Lock Register (LCK). This may be combined with CHI command “SEND_MTS”. If both bits MTSA and MTSB are set to 1 an MTS symbol will be transmitted on both channels when requested by writing SUCC1.CMD = 1000_B.

COMMAND_NOT_ACCEPTED

SUCC1.CMD is reset to 0000_B due to one of the following conditions:

- Illegal command applied by the Host
- Host writes command_not_accepted
- Host applied new command while execution of the previous Host command has not completed

When SUCC1.CMD is reset to 0000_B, bit EIR.CNA in the Error Service Request register is set, and - if enabled - an service request is generated. Commands which are not accepted are not executed.

CONFIG

Go to POC state “CONFIG” when called in POC states “DEFAULT_CONFIG“, “READY“, or in “MONITOR_MODE“. When called in “HALT” state transits to POC state

“DEFAULT_CONFIG“. When called in any other state, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”.

READY

Go to POC state “READY” when called in POC states “CONFIG”, “NORMAL_ACTIVE”, “NORMAL_PASSIVE”, “STARTUP”, or “WAKEUP”. When called in any other state, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”.

WAKEUP

Go to POC state WAKEUP when called in POC state “READY”. When called in any other state, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”.

RUN

Go to POC state “STARTUP” when called in POC state “READY”. When called in any other state, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”.

ALL_SLOTS

Leave SINGLE slot mode after successful startup / integration at the next end of cycle when called in POC states “NORMAL_ACTIVE” or “NORMAL_PASSIVE”. When called in any other state, SUCC1.CMD will be reset to
0000_B = “COMMAND_NOT_ACCEPTED”.

HALT

Set the halt request CCSV.HRQ bit in the Communication Controller Status Vector register and go to POC state “HALT” at the next end of cycle when called in POC states “NORMAL_ACTIVE” or “NORMAL_PASSIVE“. When called in any other state, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED“.

FREEZE

Set the freeze status indicator CCSV.FSI and go to POC state “HALT” immediately. Can be called from any state.

SEND_MTS

Send single MTS symbol during the next following symbol window on the channel configured by SUCC1.MTSA, SUCC1.MTSB, when called in POC state “NORMAL_ACTIVE”. When called in any other state, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”.

ALLOW_COLDSTART

The command resets bit CCSV.CSI to enable the node to become cold starter. When called in states “DEFAULT_CONFIG“, “CONFIG”, “HALT”, or “MONITOR_MODE“. SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”. To become leading coldstarter it is also required that both TXST and TXSY are set.

RESET_STATUS_INDICATORS

Resets status flags CCSV.CSNI, CCSV.CSAI, CCSV.WSV to their default values. May be called in POC state READY. When called in any other state, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”.

MONITOR_MODE

Enter MONITOR_MODE when called in POC state CONFIG. In this mode the Communication Controller is able to receive FlexRay™ Frames and wakeup pattern. It is also able to detect coding errors. The temporal integrity of received Frames is not checked. This mode can be used for debugging purposes, e.g. in case that the startup of a FlexRay™ network fails. When called in any other state, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”. For details see **“MONITOR_MODE” on Page 23-199**.

CLEAR_RAMS

Sets bit MHDS.CRAM in the Message Handler Status register when called in “DEFAULT_CONFIG” or “CONFIG” state. When called in any other state, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”. By setting MHDS.CRAM all internal RAM blocks are initialized to zero. Note that only the currently active IBF bank is cleared. To clear the 2nd bank as well, CUST1.IBF1PAG and CUST1.IBF2PAG need to be set and command CLEAR_RAMS need to be issued again. This is required in particular after an application reset. If the 2nd bank of IBF is left unused, this procedure is not required. During the initialization of the RAMs, SUCC1.PBSY will show POC busy. Access to the configuration and status registers is possible during execution of CHI command CLEAR_RAMS.

The initialization of the E-Ray internal RAM blocks requires 2048 f_{CLC_ERAY} cycles. There should be no Host access to IBF or OBF during initialization of the internal RAM blocks after application reset or after assertion of CHI command CLEAR_RAMS. Before asserting CHI command CLEAR_RAMS the Host should make sure that no transfer between Message RAM and IBF / OBF or the Transient Buffer RAMs is ongoing. This command also resets the Message Buffer Status registers MHDS, LDTS, FSR, MHDF, TXRQ1, TXRQ2, TXRQ3, TXRQ4, NDAT1, NDAT2, NDAT3, NDAT4, MBSC1, MBSC2, MBSC3, and MBSC4.

Note: All accepted commands with exception of CLEAR_RAMS and SEND_MTS will cause a change of register CCSV after at most 8 cycles of the slower of the two clocks f_{CLC_ERAY} and f_{SCLK}, assumed that POC was not busy when the command was applied and that no POC state change was forced by bus activity in that time Frame. Reading register CCSV will show data that is delayed by synchronization from f_{SCLK} to f_{CLC_ERAY} domain and by the Host-specific CPU interface.

Table 23-4 below references the CHI commands from the FlexRay™ Protocol Specification v2.1 (section 2.2.1.1, Table 2-2) to the E-Ray CHI command vector CMD.]

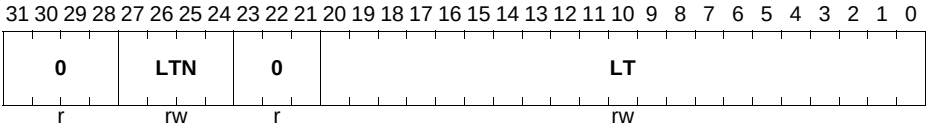
Table 23-4 Reference to CHI Host command summary from FlexRay™ protocol specification

CHI Command	Where processed (POC State)	CHI Command Vector CMD
ALL_SLOT	POC:NORMAL_ACTIVE, POC:NORMAL_PASSIVE	ALL_SLOTS
ALLOW_COLDSTART	All except POC:DEFAULT_CONFIG, POC:CONFIG, POC:HALT	ALLOW_COLDSTART
CONFIG	POC:DEFAULT_CONFIG, POC:READY	CONFIG
CONFIG_COMPLETE	POC:CONFIG	Unlock sequence & READY
DEFAULT_CONFIG	POC:HALT	CONFIG
FREEZE	All	FREEZE
HALT	POC:NORMAL_ACTIVE, POC:NORMAL_PASSIVE	HALT
READY	All except POC:DEFAULT_CONFIG, POC:CONFIG, POC:READY, POC:HALT	READY
RUN	POC:READY	RUN
WAKEUP	POC:READY	WAKEUP

SUC Configuration Register 2 (SUCC2)

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

SUCC2
SUC Configuration Register 2 (0084_H) Reset Value: 0100 0504_H



Field	Bits	Type	Description
LT	[20:0]	rw	**Listen Timeout**[1] (pdListenTimeout) Configures wakeup / startup listen timeout in Microticks. The range for wakeup / startup listen timeout (pdListenTimeout) is 1284 to 1283846 (504_H to 139706_H) Microticks
LTN	[27:24]	rw	**Listen Time-out Noise**[1] (gListenNoise - 1) Configures the upper limit for startup and wakeup listen timeout in the presence of noise expressed as a multiple of the cluster constant pdListenTimeout. The range of pdListenTimeout 2 to 16. LTN must be configured identical in all nodes of a cluster. 1_H Listen Time-out Noise is equal 2 2_H Listen Time-out Noise is equal 3 $..._H$ ... F_H Listen Time-out Noise is equal 16
0	[23:21], [31:28]	r	**Reserved** Returns 0 if read; should be written with 0.

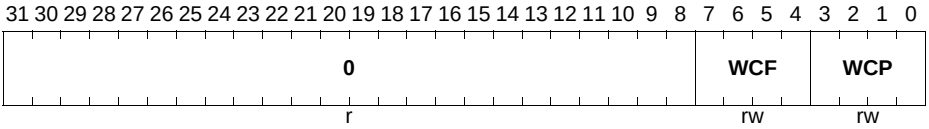
1) This bit can be updated in “DEFAULT_CONFIG” or “CONFIG” state only!

Note: The wakeup / startup noise time-out is calculated as follows:
The wakeup / startup noise time-out = pdListenTimeout • gListenNoise
= LT • (LTN+ 1)

SUC Configuration Register 3 (SUCC3)

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

SUCC3
SUC Configuration Register 3 (0088_H) Reset Value: $0000\ 0011_H$



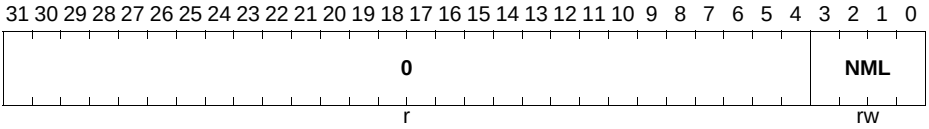
Field	Bits	Type	Description
WCP	[3:0]	rw	**Maximum Without Clock Correction Passive**[1)] (gMaxWithoutClockCorrectionPassive) Defines the number of consecutive even / odd cycle pairs with missing clock correction terms that will cause a transition from “NORMAL_ACTIVE” to “NORMAL_PASSIVE” state. Must be identical in all nodes of a cluster. Valid values are 1 to 15 (1_H to F_H) cycle pairs.
WCF	[7:4]	rw	**Maximum Without Clock Correction Fatal**[1)] (gMaxWithoutClockCorrecti on Fatal) Defines the number of consecutive even / odd cycle pairs with missing clock correction terms that will cause a transition from “NORMAL_ACTIVE” or “NORMAL_PASSIVE” to “HALT” state. Must be identical in all nodes of a cluster. Valid values are 1 to 15 (1_H to F_H)cycle pairs.
0	[31:8]	r	**Reserved** Returns 0 if read; should be written with 0.

1) This bit can be updated in “DEFAULT_CONFIG” or “CONFIG” state only!

NEM Configuration Register (NEMC)

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

NEMC
NEM Configuration Register (008C$_H$) **Reset Value: 0000 0000$_H$**



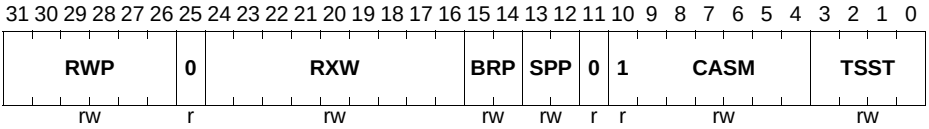
Field	Bits	Type	Description
NML	[3:0]	rw	**Network Management Vector Length**[1)] (gNetworkManagementVectorLength) These bits configure the length of the NM Vector. The configured length must be identical in all nodes of a cluster. Valid values are 0 to 12 (0_H to C_H) bytes.
0	[31:4]	r	**Reserved** Returns 0 if read; should be written with 0.

1) This bit can be updated in “DEFAULT_CONFIG” or “CONFIG” state only!

PRT Configuration Register 1 (PRTC1)

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

PRTC1
PRT Configuration Register 1 (0090_H) Reset Value: $084C\ 0633_H$



Field	Bits	Type	Description
TSST	[3:0]	rw	**Transmission Start Sequence Transmitter**[1)] (gdTSSTransmitter) Configures the duration of the Transmission Start Sequence (TSS) in terms of Bit Times (1 bit time = 4 Microticks = 100ns at 10Mbps). Must be identical in all nodes of a cluster. Valid values are 3 to 15 (3_H to F_H) Bit Times.
CASM	[10:4]	rw	**Collision Avoidance Symbol Maximum**[1)] (gdCASRxLowMax) Configures the upper limit of the acceptance window for a collision avoidance symbol (CAS). Valid values are 67 to 99 (43_H to 63_H). Most significant bit of CASM is hard wired to 1 and can not be modified.
SPP	[13:12]	rw	**Strobe Point Position**[1)] Defines the sample count value for strobing. The strobed bit value is set to the voted value when the sample count is incremented to the value configured by SPP. 00_B Sample 5 (default) 01_B Sample 4 10_B Sample 6 11_B Reserved; should not be used. *Note: The current revision 2.1 of the FlexRay™ protocol requires that SPP = 00_B. The alternate strobe point positions could be used to compensate for asymmetries in the physical layer.*

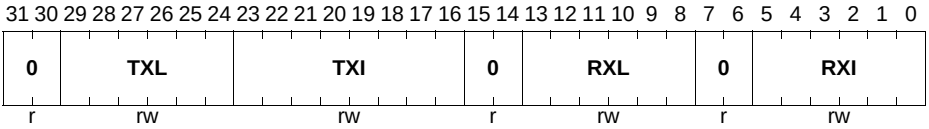
Field	Bits	Type	Description
BRP	[15:14]	rw	**Baud Rate Prescaler**[1] (gdSampleClockPeriod, pSamplePerMicrotick) The baud rate prescaler configures the baud rate on the FlexRay™ bus. The baud rates listed below are valid with a sample clock f_{SCLK}= 80 MHz. One bit time always consists of 8 samples independent of the configured baud rate. 00_B 10 Mbit/s (1 Microtick= 25 ns; twice sampled with f_{SCLK}) gdSampleClockPeriod = 12.5 ns = 1 / f_{SCLK} pSamplesPerMicrotick = 2 01_B 5 Mbit/s (1 Microtick= 25ns; single sampled with f_{SCLK} / 2) gdSampleClockPeriod = 25 ns = 2 / f_{SCLK} pSamplesPerMicrotick = 1 10_B 2.5 Mbit/s (1 Microtick = 50ns; single sampled with f_{SCLK} / 4) gdSampleClockPeriod = 50 ns = 4 / f_{SCLK} pSamplesPerMicrotick = 1 11_B Reserved; should not be used (2.5 Mbit/s (1 Microtick = 50 ns; single sampled with f_{SCLK} / 4) gdSampleClockPeriod = 50 ns = 4 / f_{SCLK} pSamplesPerMicrotick = 1
RXW	[24:16]	rw	**Wakeup Symbol Receive Window Length**[1] (gdWakeupSymbolRxWindow**)** Configures the number of Bit Times used by the node to test the duration of the received wakeup pattern. Must be identical in all nodes of a cluster. Valid values are 76 to 301 ($4C_H$ to $12D_H$) Bit Times.
RWP	[31:26]	rw	**Repetitions of Tx Wakeup Pattern**[1] (pWakeupPattern) Configures the number of repetitions (sequences) of the Tx wakeup symbol. Valid values are 2 to 63 (2_H to $3F_H$).
0	11, 25	r	**Reserved** Returns 0 if read; should be written with 0.

1) This bit can be updated in “DEFAULT_CONFIG” or “CONFIG” state only!

PRT Configuration Register 2 (PRTC2)

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

PRTC2
PRT Configuration Register 2 (0094_H) Reset Value: $0F2D\ 0A0E_H$



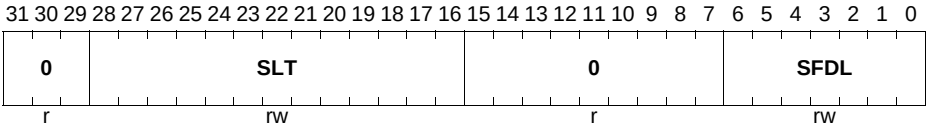
Field	Bits	Type	Description
RXI	[5:0]	rw	**Wakeup Symbol Receive Idle**[1)] (gdWakeupSymbolRxIdle) Configures the number of Bit Times used by the node to test the duration of the idle phase of the received wakeup symbol. Must be identical in all nodes of a cluster. Valid values are 14 to 59 (E_H to $3B_H$) Bit Times.
RXL	[13:8]	rw	**Wakeup Symbol Receive Low**[1)] (gdWakeupSymbolRxLow) Configures the number of Bit Times used by the node to test the duration of the low phase of the received wakeup symbol. Must be identical in all nodes of a cluster. Valid values are 10 to 55 (A_H to 37_H) Bit Times.
TXI	[23:16]	rw	**Wakeup Symbol Transmit Idle**[1)] (gdWakeupSymbolTxIdle) Configures the number of Bit Times used by the node to transmit the idle phase of the wakeup symbol. Must be identical in all nodes of a cluster. Valid values are 45 to 180 ($2D_H$ to $B4_H$)Bit Times.
TXL	[29:24]	rw	**Wakeup Symbol Transmit Low**[1)] (gdWakeupSymbolTxLow) Configures the number of Bit Times used by the node to transmit the low phase of the wakeup symbol. Must be identical in all nodes of a cluster. Valid values are 15 to 60 (F_H to $3C_H$) Bit Times.
0	[7:6], [15:14], [31:30]	r	**Reserved** Returns 0 if read; should be written with 0.

1) This bit can be updated in “DEFAULT_CONFIG” or “CONFIG” state only!

MHD Configuration Register (MHDC)

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

MHDC
MHD Configuration Register **(0098_H)** **Reset Value: $0000\ 0000_H$**



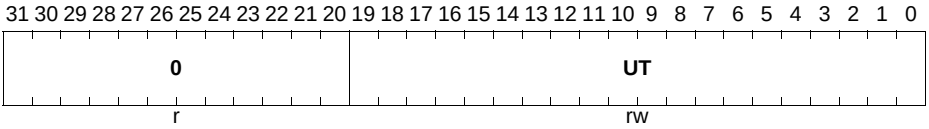
Field	Bits	Type	Description
SFDL	[6:0]	rw	**Static Frame Data Length** (gPayloadLengthStatic)[1)] Configures the cluster-wide payload length for all Frames sent in the static segment in double byte. The payload length must be identical in all nodes of a cluster. Valid values are 0 to 127 (0 to $7F_H$).
SLT	[28:16]	rw	**Start of Latest Transmit** (pLatestTx)[1)] Configures the maximum minislot value allowed before inhibiting Frame transmission in the dynamic segment of the cycle. There is no transmission dynamic segment if SLT is reset to zero. Valid values are 0 to 7981 (0 to $1F2D_H$) minislots.
0	[15:7], [31:29]	r	**Reserved** Returns 0 if read; should be written with 0.

1) This bit can be updated in “DEFAULT_CONFIG” or “CONFIG” state only!

GTU Configuration Register 1 (GTUC01)

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

GTUC01
GTU Configuration Register 1 (00A0$_H$) **Reset Value: 0000 0280$_H$**



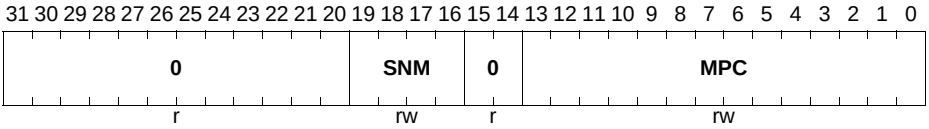
Field	Bits	Type	Description
UT	[19:0]	rw	**Microtick per Cycle** (pMicroPerCycle)[1)] Configures the duration of the communication cycle in Microticks. Valid values are 640 to 640000 (280$_H$ to 9C400$_H$) Microticks.
0	[31:20]	r	**Reserved** Returns 0 if read; should be written with 0.

1) This bit can be updated in “DEFAULT_CONFIG” or “CONFIG” state only!

GTU Configuration Register 2 (GTUC02)

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

GTUC02
GTU Configuration Register 2 (00A4$_H$) **Reset Value: 0002 000A$_H$**



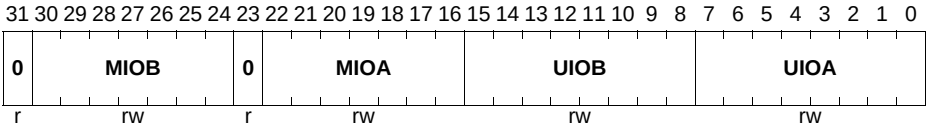
Field	Bits	Type	Description
MPC	[13:0]	rw	**Macrotick Per Cycle** (gMacroPerCycle)[1)] Configures the duration of one communication cycle in Macroticks. The cycle length must be identical in all nodes of a cluster. Valid values are 10 to 16000 (A_H to $3E80_H$) Macroticks.
SNM	[19:16]	rw	**Sync Node Max** (gSyncNodeMax)[1)] Maximum number of Frames within a cluster with SYNC Frame indicator bit SYN set to 1. Must be identical in all nodes of a cluster. Valid values are 2 to 15 (2_H to F_H).
0	[15:14], [31:20]	r	**Reserved** Returns 0 if read; should be written with 0.

1) This bit can be updated in “DEFAULT_CONFIG” or “CONFIG” state only!

GTU Configuration Register 3 (GTUC03)

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

GTUC03
GTU Configuration Register 3 (00A8$_H$) Reset Value: 0202 0000$_H$



Field	Bits	Type	Description
UIOA	[7:0]	rw	**Microtick Initial Offset Channel A**[1)] (pMicroInitialOffset[A]) Configures the number of Microticks between the actual time reference point on channel A and the subsequent Macrotick boundary of the secondary time reference point. The parameter depends on pDelayCompensation[A] and therefore has to be set for each channel independently. Valid values are 0 to 240 (0_H to $F0_H$) Microticks.
UIOB	[15:8]	rw	**Microtick Initial Offset Channel B**[1)] (pMicroInitialOffset[B]) Configures the number of Microticks between the actual time reference point on channel B and the subsequent Macrotick boundary of the secondary time reference point. The parameter depends on pDelayCompensation[B] and therefore has to be set for each channel independently. Valid values are 0 to 240 (0_H to $F0_H$) Microticks.
MIOA	[22:16]	rw	**Macrotick Initial Offset Channel A** (gMacroInitialOffset[A])[1)] Configures the number of Macroticks between the static slot boundary and the subsequent Macrotick boundary of the secondary time reference point based on the nominal Macrotick duration. Must be identical in all nodes of a cluster. Valid values are 2 to 72 (2_H to 48_H) Macroticks.
MIOB	[30:24]	rw	**Macrotick Initial Offset Channel B** (gMacroInitialOffset[B])[1)] Configures the number of Macroticks between the static slot boundary and the subsequent Macrotick boundary of the secondary time reference point based on the nominal Macrotick duration. Must be identical in all nodes of a cluster. Valid values are 2 to 72 (2_H to 48_H) Macroticks.

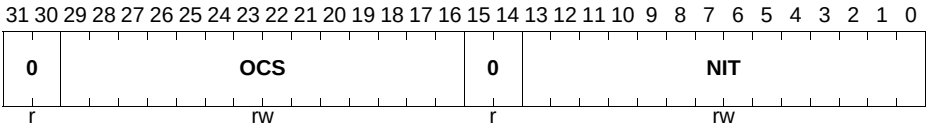
Field	Bits	Type	Description
0	23, 31	r	**Reserved** Returns 0 if read; should be written with 0.

1) This bit can be updated in “DEFAULT_CONFIG” or “CONFIG” state only!

GTU Configuration Register 4 (GTUC04)

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only. For details about configuration of NIT and OCS see **“Configuration of Network Idle Time (NIT) Start and Offset Correction Start” on Page 23-188**.

GTUC04
GTU Configuration Register 4 (00AC$_H$) Reset Value: 0008 0007$_H$



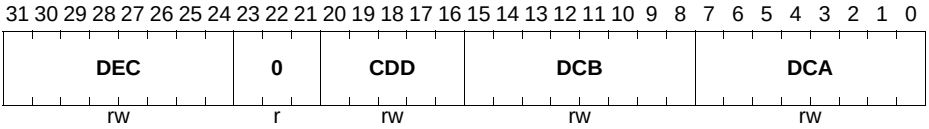
Field	Bits	Type	Description
NIT	[13:0]	rw	**Network Idle Time Start** [1)] (gMacroPerCycle - gdNIT - 1) Configures the starting point of the Network Idle Time (NIT) at the end of the communication cycle expressed in terms of Macroticks from the beginning of the cycle. The start of network idle time (NIT) is recognized if Macrotick = gMacroPerCycle - gdNIT -1 and the increment pulse of Macrotick is set. Must be identical in all nodes of a cluster. Valid values are 7 to 15997 (7_H to $3E7D_H$) Macroticks.
OCS	[29:16]	rw	**Offset Correction Start**[1)] (gOffsetCorrectionStart - 1) Determines the start of the offset correction within the network idle time (NIT) phase, calculated from start of cycle. Must be identical in all nodes of a cluster. For cluster consisting of E-Ray implementations only, it is sufficient to program OCS = NIT + 1. Valid values are 8 to 15998 (8_H to $3E7E_H$) Macroticks.
0	[15:14], [31:30]	r	**Reserved** Returns 0 if read; should be written with 0.

1) This bit can be updated in “DEFAULT_CONFIG” or “CONFIG” state only!

GTU Configuration Register 5 (GTUC05)

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

GTUC05
GTU Configuration Register 5 ($00B0_H$) Reset Value: $0E00\ 0000_H$



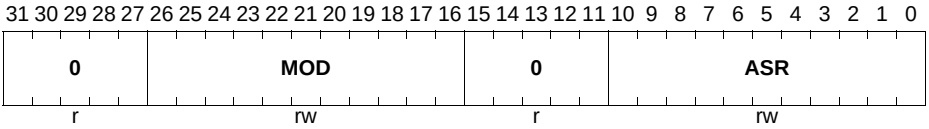
Field	Bits	Type	Description
DCA	[7:0]	rw	**Delay Compensation Channel A**[1)] (pDelayCompensation[A]) Used to compensate for reception delays on channel A. This covers assumed propagation delay up to cPropagationDelayMax for Microticks in the range of 0.0125µs to 0.05µs. In practice, the minimum of the propagation delays of all sync nodes should be applied. Valid values are 0 to 200 (0_H to $C8_H$) Microticks.
DCB	[15:8]	rw	**Delay Compensation Channel B**[1)] (pDelayCompensation[B]) Used to compensate for reception delays on channel B. This covers assumed propagation delay up to cPropagationDelayMax for Microticks in the range of 0.0125 to 0.05µs. In practice, the minimum of the propagation delays of all sync nodes should be applied. Valid values are 0 to 200 (0_H to $C8_H$) Microticks.
CDD	[20:16]	rw	**Cluster Drift Damping** (pClusterDriftDamping)[1)] Configures the cluster drift damping value used in clock synchronization to minimize accumulation of rounding errors. Valid values are 0 to 20 (0_H to 14_H) Microticks.
DEC	[31:24]	rw	**Decoding Correction** (pDecodingCorrection)[1)] Configures the decoding correction value used to determine the primary time reference point. Valid values are 14 to 143 (E_H to $8F_H$) Microticks.
0	[23:21]	r	**Reserved** Returns 0 if read; should be written with 0.

1) This bit can be updated in “DEFAULT_CONFIG” or “CONFIG” state only!

GTU Configuration Register 6 (GTUC06)

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

GTUC06
GTU Configuration Register 6 (00B4$_H$) Reset Value: 0002 0000$_H$



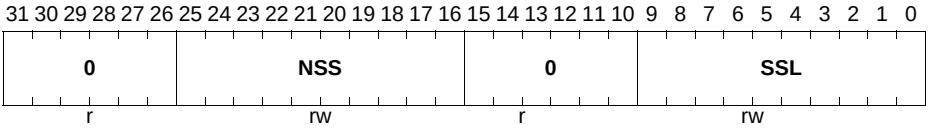
Field	Bits	Type	Description
ASR	[10:0]	rw	**Accepted Startup Range**[1] (pdAcceptedStartupRange) Number of Microticks constituting the expanded range of measured deviation for startup Frames during integration. Valid values are 0 to 1875 (0_H to 753_H) Microticks.
MOD	[26:16]	rw	**Maximum Oscillator Drift** (pdMaxDrift)[1] Maximum drift offset between two nodes that operate with unsynchronized clocks over one communication cycle in Microticks. Valid values are 2 to 1923 (2_H to 783_H) Microticks.
0	[15:11], [31:27]	r	**Reserved** Returns 0 if read; should be written with 0.

1) This bit can be updated in “DEFAULT_CONFIG” or “CONFIG” state only!

GTU Configuration Register 7 (GTUC07)

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

GTUC07
GTU Configuration Register 7 (00B8$_H$) **Reset Value: 0002 0004$_H$**



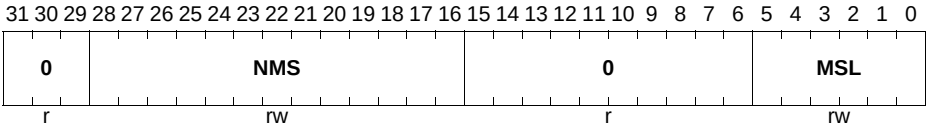
Field	Bits	Type	Description
SSL	[9:0]	rw	**Static Slot Length** (gdStaticSlot)[1)] Configures the duration of a static slot in Macroticks. The static slot length must be identical in all nodes of a cluster. Valid values are 4 to 659 (4_H to 293_H) Macroticks.
NSS	[25:16]	rw	**Number of Static Slots** (gNumberOfStaticSlots)[1)] Configures the number of static slots in a cycle. At least 2 coldstart nodes must be configured to startup a FlexRay™ network. The number of static slots must be identical in all nodes of a cluster. Valid values are 2 to 1023 (2_H to $3FF_H$).
0	[15:10], [31:26]	r	**Reserved** Returns 0 if read; should be written with 0.

1) This bit can be updated in “DEFAULT_CONFIG” or “CONFIG” state only!

GTU Configuration Register 8 (GTUC08)

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

GTUC08
GTU Configuration Register 8 (00BC$_H$) Reset Value: 0000 0002$_H$



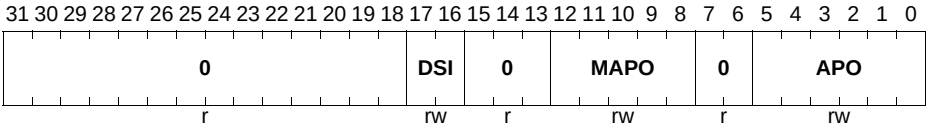
Field	Bits	Type	Description
MSL	[5:0]	rw	**Minislot Length** (gdMinislot)[1)] Configures the duration of a minislot in Macroticks. The minislot length must be identical in all nodes of a cluster. Valid values are 2 to 63 (2_H to $3F_H$) Macroticks.
NMS	[28:16]	rw	**Number of Minislots** (gNumberOfMinislots)[1)] Configures the number of minislots within the dynamic segment of a cycle. The number of minislots must be identical in all nodes of a cluster. Valid values are 0 to 7986 (0_H to $1F32_H$).
0	[15:6], [31:29]	r	**Reserved** Returns 0 if read; should be written with 0.

1) This bit can be updated in “DEFAULT_CONFIG” or “CONFIG” state only!

GTU Configuration Register 9 (GTUC09)

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

GTUC09
GTU Configuration Register 9 ($00C0_H$) Reset Value: $0000\ 0101_H$



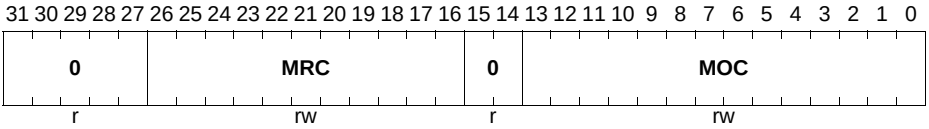
Field	Bits	Type	Description
APO	[5:0]	rw	**Action Point Offset** (gdActionPointOffset)[1)] Configures the action point offset in Macroticks within static slots and symbol window. Must be identical in all nodes of a cluster. Valid values are 1 to 63 (1_H to $3F_H$) Macroticks.
MAPO	[12:8]	rw	**Minislot Action Point Offset**[1)] (gd Minislot Action Point Offset) Configures the action point offset in Macroticks within the minislots of the dynamic segment. Must be identical in all nodes of a cluster. Valid values are 1 to 31 (1_H to $1F_H$) Macroticks.
DSI	[17:16]	rw	**Dynamic Slot Idle Phase**[1)] (gdDynamicSlotIdlePhase) The duration of the dynamic slot idle phase has to be greater or equal than the idle detection time. Must be identical in all nodes of a cluster. Valid values are 0 to 2 Minislot.
0	[7:6], [15:13], [31:18]	r	**Reserved** Returns 0 if read; should be written with 0.

1) This bit can be updated in “DEFAULT_CONFIG” or “CONFIG” state only!

GTU Configuration Register 10 (GTUC10)

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

GTUC10
GTU Configuration Register 10 ($00C4_H$) Reset Value: $0002\ 0005_H$

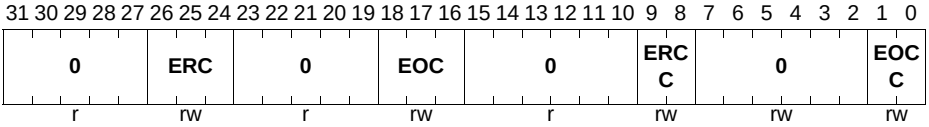


Field	Bits	Type	Description
MOC	[13:0]	rw	**Maximum Offset Correction**[1] (pOffsetCorrectionOut) Holds the maximum permitted offset correction value to be applied by the internal clock synchronization algorithm (absolute value). The Communication Controller checks only the internal offset correction value against the maximum offset correction value. Valid values are 5 to 15266 (5_H to $3BA2_H$) Microticks.
MRC	[26:16]	rw	**Maximum Rate Correction**[1] (pRateCorrectionOut) Holds the maximum permitted rate correction value to be applied by the internal clock synchronization algorithm. The communication controller checks only the internal rate correction value against the maximum rate correction value (absolute value). Valid values are 2 to 1923 (2_H to 783_H) Microticks.
0	[15:14], [31:27]	r	**Reserved** Returns 0 if read; should be written with 0.

1) This bit can be updated in “DEFAULT_CONFIG” or “CONFIG” state only!

GTU Configuration Register 11 (GTUC11)

GTUC11
GTU Configuration Register 11 (00C8$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
EOCC	[1:0]	rw	**External Offset Correction Control** (pExternOffsetControl) By writing to EOCC the external offset correction is enabled as specified below. Should be modified only outside network idle time (NIT). 00_B No external clock correction 01_B No external clock correction 10_B External offset correction value subtracted from calculated offset correction value 11_B External offset correction value added to calculated offset correction value
ERCC	[9:8]	rw	**External Rate Correction Control** (pExternRateControl) By writing to ERCC the external rate correction is enabled as specified below. Should be modified only outside network idle time (NIT). 00_B No external rate correction 01_B No external rate correction 10_B External rate correction value subtracted from calculated rate correction value 11_B External rate correction value added to calculated rate correction value
EOC	[18:16]	rw	**External Offset Correction**[1)] (pExternOffsetCorrection) Holds the external clock offset correction value in Microticks to be applied by the internal synchronization algorithm. The value is subtracted / added from / to the calculated offset correction value. The value is applied during network idle time (NIT). May be modified in “DEFAULT_CONFIG” or “CONFIG” state only. Valid values are 0 to 7 Microticks.

Field	Bits	Type	Description
ERC	[26:24]	rw	**External Rate Correction**[1] (pExternRateCorrection) Holds the external rate correction value in Microticks to be applied by the internal clock synchronization algorithm. The value is subtracted / added from / to the calculated rate correction value. The value is applied during network idle time (NIT). May be modified in “DEFAULT_CONFIG” or “CONFIG” state only. Valid values are 0 to 7 Microticks.
0	[7:2], [15:10], [23:19], [31:27]	r	**Reserved** Returns 0 if read; should be written with 0.

1) This bit can be updated in “DEFAULT_CONFIG” or “CONFIG” state only!

23.5.2.5 Communication Controller Status Registers

During 8/16-bit accesses to status variables coded with more than 8/16-bit, the variable might be updated by the Communication Controller between two accesses (non-atomic read accesses). The status vector may change faster than the Host can poll the status vector, depending on f_{CLC_ERAY} frequency.

Communication Controller Status Vector (CCSV)

CCSV

Communication Controller Status Vector(0100_H) **Reset Value: $0010\ 4000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		PSL						RCA					WSV		
r		rh						rh					rh		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CSI	CSAI	CSNI	0		SLM		HRQ	FSI	POCS					
r	rh	rh	rh	r		rh		rh	rh	rh					

Field	Bits	Type	Description
POCS	[5:0]	rh	**Protocol Operation Control Status** Indicates the actual state of operation of the Communication Controller Protocol Operation Control 000000_B “DEFAULT_CONFIG” state 000001_B “READY” state 000010_B “NORMAL_ACTIVE” state 000011_B “NORMAL_PASSIVE” state 000100_B “HALT” state 000101_B “MONITOR_MODE” state 000110_B … 001110_B are reserved. 001111_B “CONFIG” state Indicates the actual state of operation of the POC in the wakeup path 010000_B WAKEUP_STANDBY state 010001_B “WAKEUP_LISTEN” state 010010_B “WAKEUP_SEND” state 010011_B “WAKEUP_DETECT” state 010100_B … 011111_B are reserved. Indicates the actual state of operation of the POC in the startup path 100000_B “STARTUP_PREPARE” state 100001_B “COLDSTART_LISTEN” state 100010_B “COLDSTART_COLLISION_RESOLUTION state 100011_B “COLDSTART_CONSISTENCY_CHECK” state 100100_B “COLDSTART_GAP state 100101_B “COLDSTART_JOIN” State 100110_B “INTEGRATION_COLDSTART_CHECK” state 100111_B “INTEGRATION_LISTEN” state 101000_B “INTEGRATION_CONSISTENCY_CHECK” state 101001_B “INITIALIZE_SCHEDULE” state 101010_B “ABORT_STARTUP” state 101011_B “STARTUP_SUCCESS” state 101100_B … 111111_B are reserved.

Field	Bits	Type	Description
FSI	6	rh	**Freeze Status Indicator** (vPOC!Freeze**)** Indicates that the POC has entered the “HALT” state due to CHI command “FREEZE” or due to an error condition requiring an immediate POC halt. Reset by transition from “HALT” to “DEFAULT_CONFIG” state.
HRQ	7	rh	**Halt Request** (vPOC!CHIHaltRequest) Indicates that a request from the Host has been received to halt the POC at the end of the communication cycle. Reset by transition from “HALT” to “DEFAULT_CONFIG” state or when entering “READY” state.
SLM	[9:8]	rh	**Slot Mode** (vPOC!SlotMode) Indicates the actual slot mode of the POC in states READY, WAKEUP, STARTUP, NORMAL_ACTIVE, and NORMAL_PASSIVE. Default is “SINGLE”. Changes to “ALL”, depending on configuration bit SUCC1.TSM. In “NORMAL_ACTIVE” or “NORMAL_PASSIVE” state the CHI command “ALL_SLOTS” will change the slot mode from “SINGLE” over “ALL_PENDING” to “ALL”. Set to SINGLE in all other states. 00_B SINGLE 01_B Reserved 10_B ALL_PENDING 11_B ALL
CSNI	12	rh	**Coldstart Noise Indicator** (vPOC!ColdstartNoise) Indicates that the cold start procedure occurred under noisy conditions. Reset by CHI command “RESET_STATUS_INDICATORS” or by transition from “HALT” to “DEFAULT_CONFIG” state or from “READY” to “STARTUP” state.
CSAI	13	rh	**Coldstart Abort Indicator** Coldstart aborted. Reset by CHI command “RESET_STATUS_INDICATORS” or by transition from “HALT” to “DEFAULT_CONFIG” state or from “READY” to “STARTUP” state.

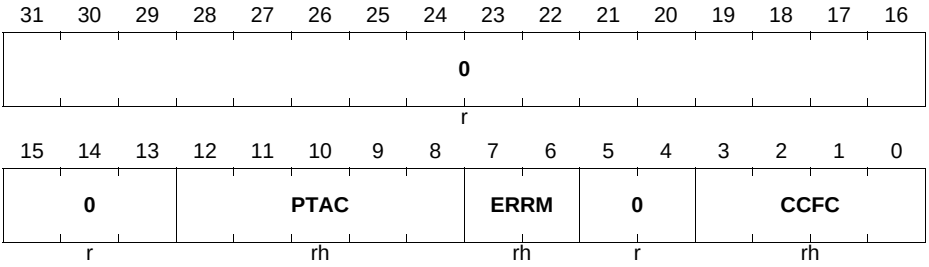
Field	Bits	Type	Description
CSI	14	rh	**Cold Start Inhibit** (vColdStartInhibit) Indicates that the node is disabled from cold starting. The flag is set whenever the POC enters “READY” state due to CHI command “READY”. The flag has to be reset under control of the Host by CHI command “ALLOW_COLDSTART” (SUCC1.CMD = 1001_B). 0_B Cold starting of node enabled 1_B Cold starting of node disabled
WSV	[18:16]	rh	**Wakeup Status** (vPOC!WakeupStatus) Indicates the status of the current wakeup attempt. Reset by CHI command “RESET_STATUS_INDICATORS” or by transition from “HALT” to “DEFAULT_CONFIG” state. 000_B UNDEFINED. Wakeup not yet executed by the Communication Controller. 001_B RECEIVED_HEADER. Set when the Communication Controller finishes wakeup due to the reception of a Frame Header without coding violation on either channel in “WAKEUP_LISTEN” state. 010_B RECEIVED_WUP. Set when the Communication Controller finishes wakeup due to the reception of a valid wakeup pattern on the configured wakeup channel in “WAKEUP_LISTEN” state. 011_B COLLISION_HEADER. Set when the Communication Controller stops wakeup due to a detected collision during wakeup pattern transmission by receiving a valid Header on either channel. 100_B COLLISION_WUP. Set when the Communication Controller stops wakeup due to a detected collision during wakeup pattern transmission by receiving a valid wakeup pattern on the configured wakeup channel. 101_B COLLISION_UNKNOWN. Set when the Communication Controller stops wakeup by leaving “WAKEUP_DETECT” state after expiration of the wakeup timer without receiving a valid wakeup pattern or a valid Frame Header. 110_B TRANSMITTED. Set when the Communication Controller has successfully completed the transmission of the wakeup pattern. 111_B Reserved

Field	Bits	Type	Description
RCA	[23:19]	rh	**Remaining Coldstart Attempts** (vRemainingColdstartAttempts) Indicates the number of remaining coldstart attempts. The RUN command resets this counter to the maximum number of coldstart attempts as configured by SUCC1.CSA.
PSL	[29:24]	rh	**POC Status Log** Status of CCSV.POCS immediately before entering “HALT” state. Set when entering “HALT” state. Set to “HALT” when FREEZE command is applied during “HALT” state. Reset to 000000_B when leaving “HALT” state.
0	[11:10], 15, [31:30]	rh	**Reserved** Returns 0 if read; should be written with 0.

Communication Controller Error Vector (CCEV)

Reset by transition from “HALT” to “DEFAULT_CONFIG” state or when entering “READY” state.

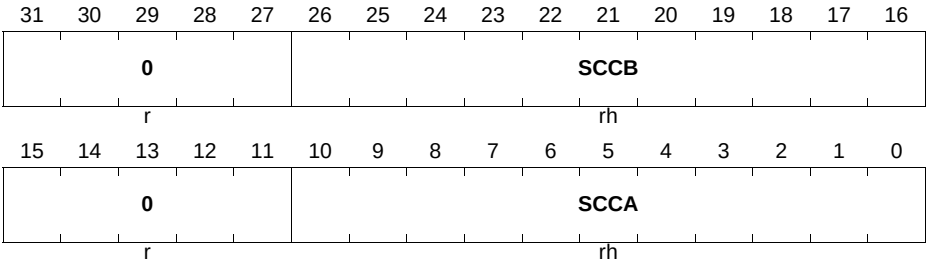
CCEV
Communication Controller Error Vector (0104_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CCFC	[3:0]	rh	**Clock Correction Failed Counter** (vClockCorrectionFailed) The Clock Correction Failed Counter is incremented by one at the end of any odd communication cycle where either the missing offset correction error or missing rate correction error are active. The Clock Correction Failed Counter is reset to 0 at the end of an odd communication cycle if neither the offset correction failed nor the rate correction failed errors are active. The Clock Correction Failed Counter stops at 15.
ERRM	[7:6]	rh	**Error Mode** (vPOC!ErrorMode) Indicates the actual error mode of the POC. 00_B “ACTIVE” (green) 01_B “PASSIVE” (yellow) 10_B “COMM_HALT” (red) 11_B Reserved
PTAC	[12:8]	rh	**Passive to Active Count** (vAllowPassiveToActive) Indicates the number of consecutive even / odd cycle pairs that have passed with valid rate and offset correction terms, while the node is waiting to transit from “NORMAL_PASSIVE” state to “NORMAL_ACTIVE” state. The transition takes place when PTAC equals SUCC1.PTA.
0	[5:4], [31:13]	r	**Reserved** Returns 0 if read; should be written with 0.

Slot Counter Value (SCV)

SCV
Slot Counter Value **(0110_H)** **Reset Value: $0000\ 0000_H$**

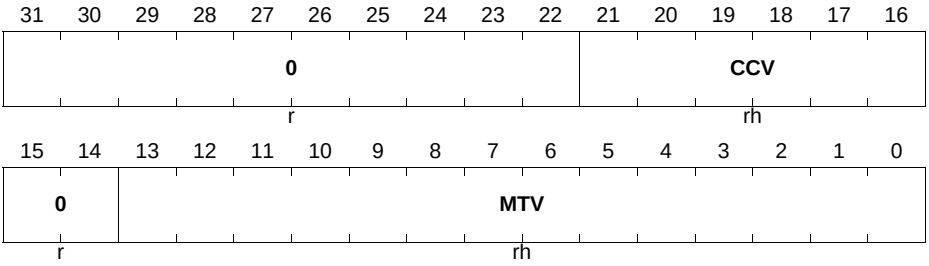


Field	Bits	Type	Description
SCCA	[10:0]	rh	**Slot Counter Channel A** (vSlotCounter[A]**)** Current slot counter value on channel A. The value is incremented by the Communication Controller and reset at the start of a communication cycle. Valid values are 0 to 2047 (0_H to $7FD_H$).
SCCB	[26:16]	rh	**Slot Counter Channel B** (vSlotCounter[B]**)** Current slot counter value on channel B. The value is incremented by the Communication Controller and reset at the start of a communication cycle. Valid values are 0 to 2047 (0_H to $7FD_H$).
0	[15:11], [31:27]	r	**Reserved** Returns 0 if read; should be written with 0.

Macrotick and Cycle Counter Value (MTCCV)

MTCCV

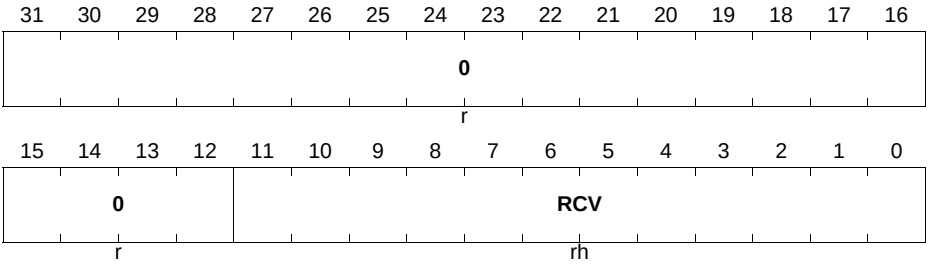
Macrotick and Cycle Counter Value (0114_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
MTV	[13:0]	rh	**Macrotick Value** (vMacrotick) Current Macrotick value. The value is incremented by the Communication Controller and reset at the start of a communication cycle. Valid values are 0 to 16000 (0_H to $3E80_H$).
CCV	[21:16]	rh	**Cycle Counter Value** (vCycleCounter) Current cycle counter value. The value is incremented by the Communication Controller at the start of a communication cycle. Valid values are 0 to 63 (0_H to $3F_H$).
0	[15:14], [31:22]	r	**Reserved** Returns 0 if read; should be written with 0.

Rate Correction Value (RCV)

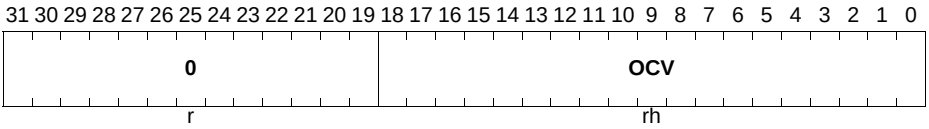
RCV
Rate Correction Value **(0118_H)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RCV	[11:0]	rh	**Rate Correction Value** (vRateCorrection) Rate correction value (two's complement). Calculated internal rate correction value before limitation. If the RCV value exceeds the limits defined by GTUC10.MRC, flag SFS.RCLR is set to 1.
0	[31:12]	r	**Reserved** Returns 0 if read; should be written with 0.

Offset Correction Value (OCV)

OCV
Offset Correction Value **($011C_H$)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
OCV	[18:0]	rh	**Offset Correction Value** (vOffsetCorrection) Offset correction value (two's complement). Calculated internal offset correction value before limitation. If the OCV value exceeds the limits defined by GTUC10.MOC flag SFS.OCLR is set to 1.
0	[31:19]	r	**Reserved** Returns 0 if read; should be written with 0.

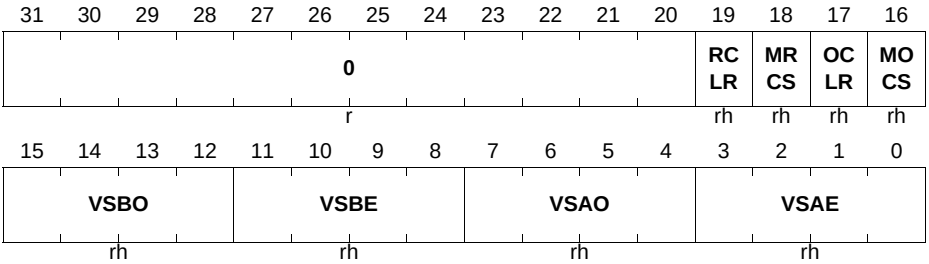
Note: The external rate / offset correction value is added to the limited rate / offset correction value.

SYNC Frame Status (SFS)

The maximum number of valid SYNC Frames in a communication cycle is 15.

SFS

SYNC Frame Status **(0120_H)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
VSAE	[3:0]	rh	**Valid SYNC Frames Channel A, even communication cycle** Holds the number of valid SYNC Frames received on channel A in the even communication cycle. If transmission of SYNC Frames is enabled by SUCC1.TXSY the value is incremented by one. The value is updated during the network idle time (NIT) of each even communication cycle. This bit field is only valid if the channel A is assigned to the Communication Controller by SUCC1.CCHA.
VSAO	[7:4]	rh	**Valid SYNC Frames Channel A, odd communication cycle** Holds the number of valid SYNC Frames received on channel A in the odd communication cycle. If transmission of SYNC Frames is enabled by SUCC1.TXSY the value is incremented by one. The value is updated during the network idle time (NIT) of each odd communication cycle. This bit field is only valid if the channel A is assigned to the Communication Controller by SUCC1.CCHA.
VSBE	[11:8]	rh	**Valid SYNC Frames Channel B, even communication cycle** Holds the number of valid SYNC Frames received on channel B in the even communication cycle. If transmission of SYNC Frames is enabled by SUCC1.TXSY the value is incremented by one. The value is updated during the network idle time (NIT) of each even communication cycle. This bit field is only valid if the channel B is assigned to the Communication Controller by SUCC1.CCHB.

Field	Bits	Type	Description
VSBO	[15:12]	rh	**Valid SYNC Frames Channel B, odd communication cycle** Holds the number of valid SYNC Frames received on channel B in the odd communication cycle. If transmission of SYNC Frames is enabled by SUCC1.TXSY the value is incremented by one. The value is updated during the network idle time (NIT) of each odd communication cycle. This bit field is only valid if the channel B is assigned to the Communication Controller by SUCC1.CCHB.
MOCS	16	rh	**Missing Offset Correction Signal** The Missing Offset Correction flag signals to the Host, that no offset correction calculation can be performed because no SYNC Frames were received. The flag is updated by the Communication Controller at start of offset correction phase. 0_B Offset correction signal valid 1_B Missing offset correction signal
OCLR	17	rh	**Offset Correction Limit Reached** The Offset Correction Limit Reached flag signals to the Host, that the offset correction value has exceeded its limit as defined by GTUC10.MOC. The flag is updated by the Communication Controller at start of offset correction phase. 0_B Offset correction below limit 1_B Offset correction limit reached
MRCS	18	rh	**Missing Rate Correction Signal** The Missing Rate Correction Flag signals to the Host, that no rate correction calculation can be performed because no pairs of even / odd SYNC Frames were received. The flag is updated by the Communication Controller at start of offset correction phase. 0_B Rate correction signal valid 1_B Missing rate correction signal
RCLR	19	rh	**Rate Correction Limit Reached** The Rate Correction Limit Reached flag signals to the Host, that the rate correction value has exceeded its limit.as defined by GTUC10.MRC. The flag is updated by the Communication Controller at start of offset correction phase. 0_B Rate correction below limit 1_B Rate correction limit reached
0	[31:20]	r	**Reserved** Returns 0 if read; should be written with 0.

Symbol Window and network idle time (NIT) Status (SWNIT)

Symbol window related status information. Updated by the Communication Controller at the end of the symbol window for each channel. During startup the status data is not updated.

Note: MTSA and MTSB may be changed outside "DEFAULT_CONFIG" or "CONFIG" state when the write to SUC Configuration Register 1 (SUCC1) register is directly preceded by the unlock sequence as described in ***"Lock Register (LCK)" on Page 23-30****. This may be combined with CHI command SEND_MTS. If both bits MTSA and MTSB are set to 1 an MTS symbol will be transmitted on both channels when requested by writing SUCC1.CMD = 1000_B*

SWNIT
Symbol Window and Network Idle Time Status
(0124_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				SBNB	SENB	SBNA	SENA	MTSB	MTSA	TCSB	SBSB	SESB	TCSA	SBSA	SESA
r				rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
SESA	0	rh	**Syntax Error in Symbol Window Channel A** (vSS!SyntaxErrorA) 0_B No syntax error detected 1_B Syntax error during symbol window detected on channel A
SBSA	1	rh	**Slot Boundary Violation in Symbol Window Channel A** (vSS!BViolationA) 0_B No slot boundary violation detected 1_B Slot boundary violation during symbol window detected on channel A

Field	Bits	Type	Description
TCSA	2	rh	**Transmission Conflict in Symbol Window Channel A** (vSS!TxConflictA) 0_B No transmission conflict detected 1_B Transmission conflict in symbol window detected on channel A
SESB	3	rh	**Syntax Error in Symbol Window Channel B** (vSS!SyntaxErrorB) 0_B No syntax error detected 1_B Syntax error during symbol window detected on channel B
SBSB	4	rh	**Slot Boundary Violation in Symbol Window Channel B** (vSS!BViolationB) 0_B No slot boundary violation detected 1_B Slot boundary violation during symbol window detected on channel B
TCSB	5	rh	**Transmission Conflict in Symbol Window Channel B** (vSS!TxConflictB) 0_B No transmission conflict detected 1_B Transmission conflict in symbol window detected on channel B
MTSA	6	rh	**MTS Received on Channel A** (vSS!ValidMTSA)[1)] Media Access Test symbol received on channel A during the proceeding symbol window. Updated by the Communication Controller for each channel at the end of the symbol window. When this bit is set to 1, also interrupt flag SIR.MTSA is set to 1. 0_B No MTS symbol received on channel A 1_B MTS symbol received on channel A
MTSB	7	rh	**MTS Received on Channel B** (vSS!ValidMTSB)[1)] Media Access Test symbol received on channel B during the proceeding symbol window. Updated by the Communication Controller for each channel at the end of the symbol window. When this bit is set to 1, also interrupt flag SIR.MTSB is set to 1. 0_B No MTS symbol received on channel B 1_B MTS symbol received on channel B

Field	Bits	Type	Description
SENA	8	rh	**Syntax Error during network idle time (NIT) Channel A** (vSS!SyntaxErrorA) Updated by the Communication Controller channel A at the end of the NIT. 0_B No syntax error detected 1_B Syntax error during network idle time (NIT) detected on channel A
SBNA	9	rh	**Slot Boundary Violation during network idle time (NIT) Channel A** (vSS!BViolationA) Updated by the Communication Controller channel A at the end of the NIT. 0_B No slot boundary violation detected 1_B Slot boundary violation during network idle time (NIT) detected on channel A
SENB	10	rh	**Syntax Error during network idle time (NIT) Channel B** (vSS!SyntaxErrorB) Updated by the Communication Controller channel B at the end of the NIT. 0_B No syntax error detected 1_B Syntax error during network idle time (NIT) detected on channel B
SBNB	11	rh	**Slot Boundary Violation during network idle time (NIT) Channel B** (vSS!BViolationB) Updated by the Communication Controller channel B at the end of the NIT. 0_B No slot boundary violation detected 1_B Slot boundary violation during network idle time (NIT) detected on channel B
0	[31:12]	r	**Reserved** Returns 0 if read; should be written with 0.

1) MTSA and MTSB may also be changed outside “DEFAULT_CONFIG” or “CONFIG” state when the write to SUCC1 register is directly preceded by the unlock sequence as described in “Lock Register (LCK)”. This may be combined with CHI command SEND_MTS. If both bits MTSA and MTSB are set to 1 an MTS symbol will be transmitted on both channels when requested by writing SUCC1.CMD = 1000_B.

Aggregated Channel Status (ACS)

The aggregated channel status provides the Host with an accrued status of channel activity for all communication slots regardless of whether they are assigned for transmission or subscribed for reception. The aggregated channel status also includes status data from the symbol window and the network idle time. The status data is updated (set) after each slot and aggregated until it is reset by the Host. During startup the status data is not updated. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect on the flag. An application reset will also clear the register.

ACS
Aggregated Channel Status (0128$_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			SBVB	CIB	CEDB	SEDB	VFRB	0			SBVA	CIA	CEDA	SEDA	VFRA
r			rwh	rwh	rwh	rwh	rwh	r			rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
VFRA	0	rwh	**Valid Frame Received on Channel A** (vSS!ValidFrameA) One or more valid Frames were received on channel A in any static or dynamic slot during the observation period. 0_B No valid Frame received 1_B Valid Frame(s) received on channel A
SEDA	1	rwh	**Syntax Error Detected on Channel A** (vSS!SyntaxErrorA) One or more syntax errors in static or dynamic slots, symbol window, and network idle time (NIT) were observed on channel A. 0_B No syntax error observed 1_B Syntax error(s) observed on channel A
CEDA	2	rwh	**Content Error Detected on Channel A** (vSS!ContentErrorA) One or more Frames with a content error were received on channel A in any static or dynamic slot during the observation period. 0_B No Frame with content error received 1_B Frame(s) with content error received on channel A

Field	Bits	Type	Description
CIA	3	rwh	**Communication Indicator Channel A** One or more valid Frames were received on channel A in slots that also contained any additional communication during the observation period, i.e. one or more slots received a valid Frame AND had any combination of either syntax error OR content error OR slot boundary violation. 0_B No valid Frame(s) received in slots containing any additional communication 1_B Valid Frame(s) received on channel A in slots containing any additional communication
SBVA	4	rwh	**Slot Boundary Violation on Channel A** (vSS!BViolationA) One or more slot boundary violations were observed on channel A at any time during the observation period (static or dynamic slots, symbol window, and network idle time NIT). 0_B No slot boundary violation observed 1_B Slot boundary violation(s) observed on channel A
VFRB	8	rwh	**Valid Frame Received on Channel B** (vSS!ValidFrameB) One or more valid Frames were received on channel B in any static or dynamic slot during the observation period. 0_B No valid Frame received 1_B Valid Frame(s) received on channel B
SEDB	9	rwh	**Syntax Error Detected on Channel B** (vSS!SyntaxErrorB) One or more syntax errors in static or dynamic slots, symbol window, and network idle time (NIT) were observed on channel B. 0_B No syntax error observed 1_B Syntax error(s) observed on channel B
CEDB	10	rwh	**Content Error Detected on Channel B** (vSS!ContentErrorB) One or more Frames with a content error were received on channel B in any static or dynamic slot during the observation period. 0_B No Frame with content error received 1_B Frame(s) with content error received on channel B

Field	Bits	Type	Description
CIB	11	rwh	**Communication Indicator Channel B** One or more valid Frames were received on channel B in slots that also contained any additional communication during the observation period, i.e. one or more slots received a valid Frame AND had any combination of either syntax error OR content error OR slot boundary violation. 0_B No valid Frame(s) received in slots containing any additional communication 1_B Valid Frame(s) received on channel B in slots containing any additional communication
SBVB	12	rwh	**Slot Boundary Violation on Channel B** (vSS!BViolationB) One or more slot boundary violations were observed on channel B at any time during the observation period (static or dynamic slots, symbol window, and network idle time NIT). 0_B No slot boundary violation observed 1_B Slot boundary violation(s) observed on channel B
0	[7:5], [31:13]	r	**Reserved** Returns 0 if read; should be written with 0.

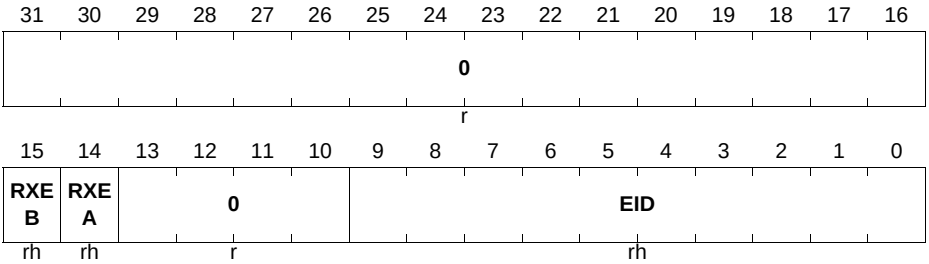
Note: The set condition of flags CIA and CIB is also fulfilled if there is only one single Frame in the slot and the slot boundary at the end of the slot is reached during the Frames channel idle recognition phase. When one of the flags SEDB, CEDB, CIB, SBVB changes from 0 to 1, service request flag EIR.EDB is set to 1.When one of the flags SEDA, CEDA, CIA, SBVA changes from 0 to 1, service request flag EIR.EDA is set to 1.

Even Sync ID [01…15] (ESIDnn)

Registers Even Sync ID nn (ESIDnn, nn=01-15) hold the Frame IDs of the SYNC Frames received in **even** communication cycles, sorted in ascending order, with register ESID01 holding the lowest received SYNC Frame ID. If the node itself transmits a SYNC Frame in an even communication cycle, register ESID01 holds the respective SYNC Frame ID as configured in Message Buffer 0 and the flags RXEA, RXEB are set. The value is updated during the network idle time (NIT) of each even communication cycle.

ESIDnn (nn = 01-15)
Even Sync ID Symbol Window nn

($012C_H$ + nn * 4) **Reset Value: 0000 0000_H**



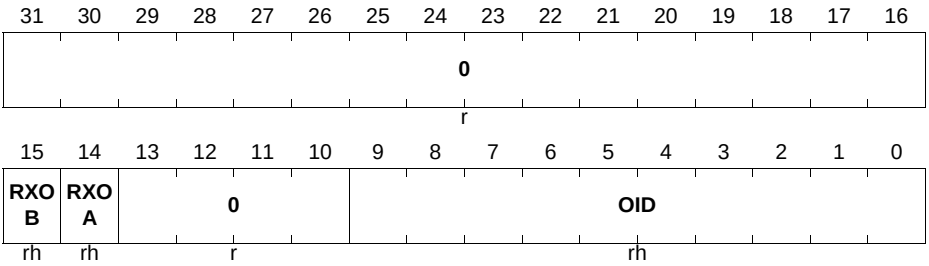
Field	Bits	Type	Description
EID	[9:0]	rh	**Even Sync ID** (vsSyncIDListA,B even) SYNC Frame ID even communication cycle.
RXEA	14	rh	**Received/Configured Even Sync ID on Channel A** Signals that a SYNC Frame corresponding to the stored even sync ID was received on channel A or that the node is configured to be a sync node with key slot = EID (ESID1 only). 0_B SYNC Frame not received on channel A / node configured to transmit SYNC Frames 1_B SYNC Frame received on channel A/ node not configured to transmit SYNC Frames
RXEB	15	rh	**Received/Configured Even Sync ID on Channel B** Signals that a SYNC Frame corresponding to the stored even sync ID was received on channel B or that the node is configured to be a sync node with key slot = EID (ESID1 only). 0_B SYNC Frame not received on channel B / node configured to transmit SYNC Frames 1_B SYNC Frame received on channel B / node not configured to transmit SYNC Frames

Field	Bits	Type	Description
0	[13:10], [31:16]	r	**Reserved** Returns 0 if read; should be written with 0.

Odd Sync ID [01…15] (OSIDnn)

he Odd Sync ID nn (OSIDnn, nn=01-15) hold the Frame IDs of the SYNC Frames received in **odd** communication cycles, sorted in ascending order, with register OSID01 holding the lowest received SYNC Frame ID. If the node itself transmits a SYNC Frame in an odd communication cycle, register OSID01 holds the respective SYNC Frame ID as configured in Message Buffer 0 and flags RXOA, RXOB are set. The value is updated during the network idle time (NIT) of each odd communication cycle.

OSIDnn (nn = 01-15)
Odd Sync ID Symbol Window nn ($016C_H$ + nn * 4) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
OID	[9:0]	rh	**Odd Sync ID** (vsSyncIDListA,B odd) SYNC Frame ID even communication cycle.
RXOA	14	rh	**Received Odd Sync ID on Channel A** Signals that a SYNC Frame corresponding to the stored odd sync ID was received on channel A or that the node is configured to be a sync node with key slot = OID (OSID1 only). 0_B SYNC Frame not received on channel A/ node configured to transmit SYNC Frames 1_B SYNC Frame received on channel A/ node not configured to transmit SYNC Frames
RXOB	15	rh	**Received Odd Sync ID on Channel B** Signals that a SYNC Frame corresponding to the stored odd sync ID was received on channel B or that the node is configured to be a sync node with key slot = OID (OSID1 only) 0_B SYNC Frame not received on channel B/ node configured to transmit SYNC Frames 1_B SYNC Frame received on channel B/ node not configured to transmit SYNC Frames

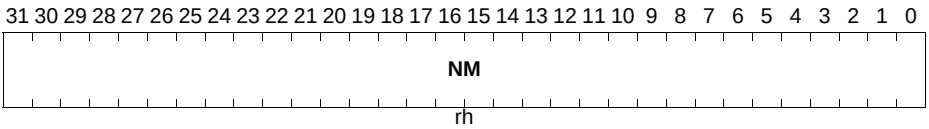
Field	Bits	Type	Description
0	[13:10], [31:16]	r	**Reserved** Returns 0 if read; should be written with 0.

Network Management Vector [1…3] (NMVx)

The three Network Management Vectors n (NMVx, x=1-3) registers hold the accrued Network Management (NM) vector (configurable 0 to 12 byte). The accrued Network Management (NM) vector is generated by the Communication Controller by bit-wise ORing each Network Management (NM) vector received (valid static Frames with PPI = 1) on each channel (see **"Network Management" on Page 23-213**). The Communication Controller updates the Network Management (NM) vector at the end of each communication cycle as long as the Communication Controller is either in "NORMAL_ACTIVE" or "NORMAL_PASSIVE" state. NMVx-bytes exceeding the configured Network Management (NM) vector length are not valid.

NMVx (x = 1-3)
Network Management Vector x

($01AC_H$ + x * 4) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
NM	[31:0]	rh	**Network Management Vector**

Table 23-5 below shows the assignment of the received payload's data byte to the Network Management vector.

Table 23-5 Assignment of Data Byte to Network Management Vector

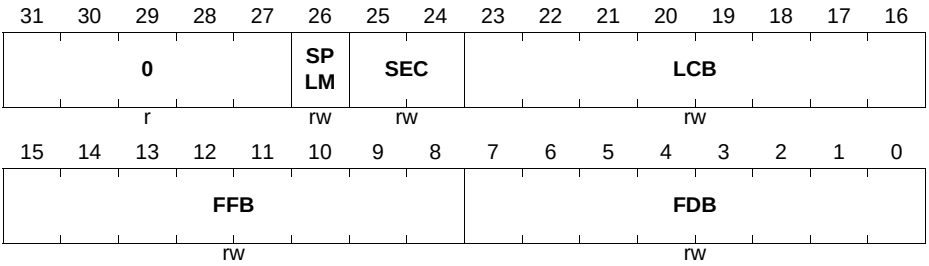
Bit / Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NM1	Data3								Data2								Data1								Data0							
NM2	Data7								Data6								Data5								Data4							
NM3	Data11								Data10								Data9								Data8							

23.5.2.6 Message Buffer Control Registers

Message RAM Configuration (MRC)

The Message RAM Configuration register defines the number of Message Buffers assigned to the static segment, dynamic segment, and FIFO. The register can be written during “DEFAULT_CONFIG” or “CONFIG” state only.

MRC
Message RAM Configuration ((0300_H)) Reset Value: $0180\ 0000_H$



Field	Bits	Type	Description
FDB	[7:0]	rw	**First Dynamic Buffer** May be modified in “DEFAULT_CONFIG” or “CONFIG” state only. 00_H No group of Message Buffers exclusively for the static segment configured 01_H …$7F_H$ Message Buffers 0 to FDB-1 reserved for static segment 80_H …FF_H No dynamic Message Buffers configured
FFB	[15:8]	rw	**First Buffer of FIFO** May be modified in “DEFAULT_CONFIG” or “CONFIG” state only. 00_H …$7E_H$ Message Buffers from FFB to LCB assigned to the FIFO $7F_H$ All Message Buffers assigned to the FIFO 80_H …FF_H No Message Buffers assigned to the FIFO

Field	Bits	Type	Description
LCB	[23:16]	rw	**Last Configured Buffer** May be only modified in “DEFAULT_CONFIG” or “CONFIG” state. 01_H …$7F_H$ Number of Message Buffers is LCB + 1 80_H …FF_H No Message Buffer configured
SEC	[25:24]	rw	**Secure Buffers** Not evaluated when the Communication Controller is in “DEFAULT_CONFIG” or “CONFIG” state. For temporary unlocking see **“Host Handling of Errors” on Page 23-248**. 00_B Reconfiguration of Message Buffers enabled with numbers < FFB enabled. *Note: In nodes configured for SYNC Frame transmission or for single slot mode operation Message Buffer 0 (and if SPLM = 1, also Message Buffer 1) Reconfiguration of all Message Buffers is always locked* 01_B Reconfiguration of Message Buffers with numbers < FDB and with numbers ≥ FFB locked and transmission of Message Buffers for static segment with numbers ≥ FDB disabled 10_B Reconfiguration of all Message Buffers locked 11_B Reconfiguration of all Message Buffers locked and transmission of Message Buffers for static segment with numbers ≥ FDB disabled
SPLM	26	rw	**SYNC Frame Payload Multiplex** This bit is only evaluated if the node is configured as sync node (SUCC1.TXSY = 1) or for single slot mode operation (SUCC1.TSM = 1). When this bit is set to 1 Message Buffers 0 and 1 are dedicated for SYNC Frame transmission with different payload data on channel A and B. When this bit is reset to 0, SYNC Frames are transmitted from Message Buffer 0 with the same payload data on both channels. Note that the channel filter configuration for Message Buffer 0 resp. Message Buffer 1 has to be chosen accordingly. 0_B Only Message Buffer 0 locked against reconfiguration 1_B Both Message Buffers 0 and 1 are locked against reconfiguration

Field	Bits	Type	Description
0	[31:27]	r	**Reserved** Returns 0 if read; should be written with 0.

*Note: In case the node is configured as sync node (*SUCC1.*TXSY = 1) or for single slot mode operation (*SUCC1.*TSM = 1), Message Buffer 0 resp. 1 is reserved for SYNC Frames or single slot Frames and have to be configured with the node-specific key slot ID. In case the node is neither configured as sync node nor for single slot operation Message Buffer 0 resp. 1 is treated like all other Message Buffers.*

Table 23-6 Usage of the three Message Buffer Pointer

Message Buffer 0	⇓ Static Buffers		
Message Buffer 1			
	⇓ Static + Dynamic	⇐ **FDB**	
	Buffers		
…			FIFO configured: **FFB** > **FDB**
	⇓ FIFO	⇐ **FFB**	No FIFO configured: **FFB** ≥ 128
Message Buffer N-1			**LCB** ≥ **FDB**, **LCB** ≥ **FFB**
Message Buffer N		⇐ **LCB**	

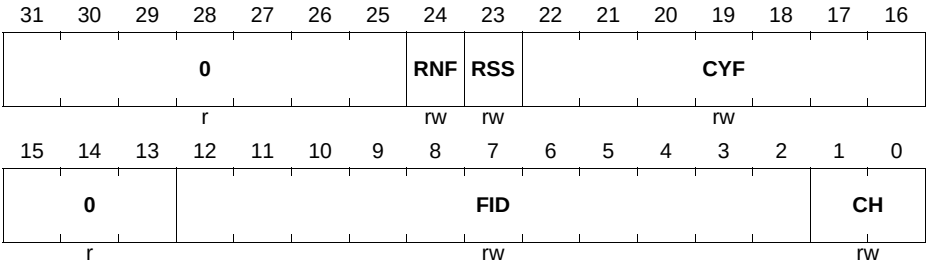
The programmer has to ensure that the configuration defined by FDB, FFB, and LCB is valid. **The Communication Controller does not check for erroneous configurations!**

Note: The maximum number of Header Sections is 128. This means a maximum of 128 Message Buffer can be configured. The maximum length of a Data Section is 254 byte. The length of the Data Section may be configured differently for each Message Buffer. For details see ***"Message RAM" on Page 23-239****.*
In case two or more Message Buffers are assigned to slot 1 by use of cycle filtering, all of them must be located either in the "Static Buffers" or at the beginning of the "Static + Dynamic Buffers" section.
The payload length configured and the length of the Data Section need to be configured identically for all Message Buffers belonging to the FIFO via WRHS2.PLC and WRHS3.DP. When the Communication Controller is not in "DEFAULT_CONFIG" or "CONFIG" state reconfiguration of Message Buffers belonging to the FIFO is locked.

FIFO Rejection Filter (FRF)

The FIFO Rejection Filter defines a user specified sequence of bits to which channel, Frame ID, and cycle count of the incoming Frames are compared. Together with the FIFO Rejection Filter Mask this register determines whether a message is rejected by the FIFO. The FRF register can be written during “DEFAULT_CONFIG” or “CONFIG” state only.

FRF
FIFO Rejection Filter **(0304_H)** **Reset Value: $0180\ 0000_H$**



Field	Bits	Type	Description
CH	[1:0]	rw	**Channel Filter** May be modified in “DEFAULT_CONFIG” or “CONFIG” state only. 00_B receive on both channels[1)] 01_B receive only on channel B 10_B receive only on channel A 11_B no reception
FID	[12:2]	rw	**Frame ID Filter** Determines the Frame ID to be rejected by the FIFO. With the additional configuration of register FRFM, the corresponding Frame ID filter bits are ignored, which results in further rejected Frame IDs. When FRFM.MFID is zero, a Frame ID filter value of zero means that no Frame ID is rejected. 000_H …$7FF_H$ Frame ID filter values

Field	Bits	Type	Description
CYF	[22:16]	rw	**Cycle Counter Filter** The 7-bit cycle counter filter determines the cycle set to which Frame ID and channel rejection filter are applied. In cycles **not** belonging to the cycle set specified by CYF, all Frames are rejected. For details about the configuration of the cycle counter filter see **“Cycle Counter Filtering” on Page 23-215**. May be modified in “DEFAULT_CONFIG” or “CONFIG” state only.
RSS	23	rw	**Reject in Static Segment** If this bit is set, the FIFO is used only be used in dynamic segment. May be modified in “DEFAULT_CONFIG” or “CONFIG” state only. 0_B FIFO also used in static segment 1_B Reject messages for static segment
RNF	24	rw	**Reject NULL Frames** If this bit is set, received NULL Frames are not stored in the FIFO. May be modified in “DEFAULT_CONFIG” or “CONFIG” state only. 0_B NULL Frames are stored in the FIFO 1_B Reject all NULL Frames
0	[15:13], [31:25]	r	**Reserved** Returns 0 if read; should be written with 0.

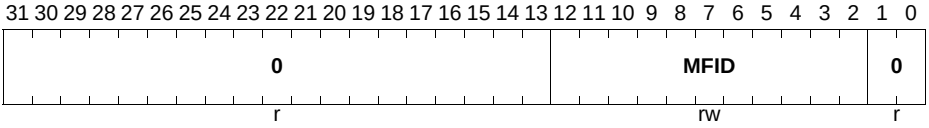
1) If reception on both channels is configured, also in static segment always both Frames (from channel A and B) are stored in the FIFO, even if they are identical.

FIFO Rejection Filter Mask (FRFM)

The FIFO Rejection Filter Mask specifies which of the corresponding Frame ID filter bits are relevant for rejection filtering. If a bit is set, it indicates that the corresponding bit in the FRF register will not be considered for rejection filtering. The FRFM register can be written during “DEFAULT_CONFIG” or “CONFIG” state only.

FRFM

FIFO Rejection Filter Mask **(0308_H)** **Reset Value: $0000\ 0000_H$**

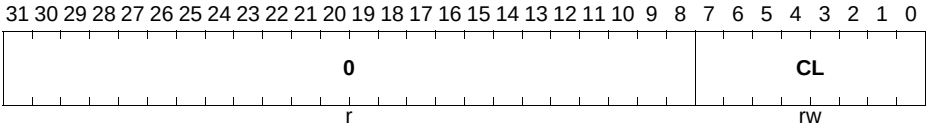


Field	Bits	Type	Description
MFID	[12:2]	rw	**Mask Frame ID Filter** May be modified in “DEFAULT_CONFIG” or “CONFIG” state only. 0_B Corresponding Frame ID filter bit is used for rejection filtering. 1_B Ignore corresponding Frame ID filter bit.
0	[1:0], [31:13]	r	**Reserved** Returns 0 if read; should be written with 0.

FIFO Critical Level (FCL)

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

FCL
FIFO Critical Level **($030C_H$)** **Reset Value: $0000\ 0080_H$**



Field	Bits	Type	Description
CL	[7:0]	rw	**Critical Level** When the receive FIFO fill level FSR.RFFL is equal or greater than the critical level configured by CL, the receive FIFO critical level flag FSR.RFCL is set. If CL is programmed to values > 128, bit FSR.RFCL is never set. When FSR.RFCL changes from 0 to 1 bit **SIR.RFCL** is set to 1, and if enabled, a service request is generated.
0	[31:8]	r	**Reserved** Returns 0 if read; should be written with 0.

23.5.2.7 Message Buffer Status Registers

Message Handler Status (MHDS)

The Message Handler Status register gives the Host access to the current state of the Message Handler. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect on the flag. An application reset will also clear the register. If one of the flags MHDS.EIBF, MHDS.EOBF, MHDS.EMR, MHDS.ETBF1, MHDS.ETBF2 changes from 0 to 1 EIR.EERR is set.

MHDS
Message Handler Status **(0310_H)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				MBU				0			MBT				
r				rh				r				rh			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				FMB				CRAM	MFMB	FMBD	ETBF2	ETBF1	EMR	EOBF	EIBF
r				rh				rh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

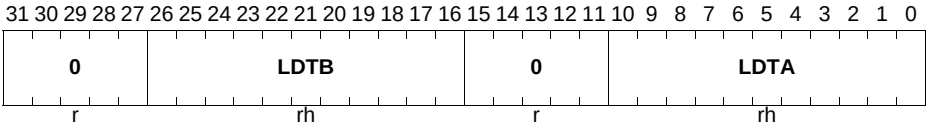
Field	Bits	Type	Description
EIBF	0	rwh	**ECC Error Input Buffer RAM 1,2** 0_B No error 1_B Error occurred when reading Input Buffer RAM 1 or Input Buffer RAM 2
EOBF	1	rwh	**ECC Error Output Buffer RAM 1,2** 0_B No error 1_B Error occurred when reading Output Buffer RAM 1 or Output Buffer RAM 2
EMR	2	rwh	**ECC Error Message RAM** 0_B No error 1_B Error occurred when reading the Message RAM
ETBF1	3	rwh	**ECC Error Transient Buffer RAM A** 0_B No error 1_B Error occurred when reading Transient Buffer RAM A
ETBF2	4	rwh	**ECC Error Transient Buffer RAM B** 0_B No error 1_B Error occurred when reading Transient Buffer RAM B

Field	Bits	Type	Description
FMBD	5	rwh	**Faulty Message Buffer Detected** 0_B No faulty Message Buffer 1_B Message Buffer referenced by MHDS.FMB holds faulty data due to a ECC error
MFMB	6	rwh	**Multiple Faulty Message Buffers detected** 0_B No additional faulty Message Buffer 1_B Another faulty Message Buffer was detected while flag MHDS.FMBD is set
CRAM	7	rh	**Clear all internal RAM's** Signals that execution of the CHI command CLEAR_RAMS is ongoing (all bits of all internal RAM blocks are written to 0). The bit is set by CHI command CLEAR_RAMS. 0_B No execution of the CHI command CLEAR_RAMS 1_B Execution of the CHI command CLEAR_RAMS ongoing
FMB	[14:8]	rh	**Faulty Message Buffer** ECC error occurred when reading from the Message Buffer or when transferring data from Input Buffer or Transient Buffer A or Transient Buffer B to the Message Buffer referenced by MHDS.FMB. Value only valid when one of the flags MHDS.EIBF, MHDS.EMR, MHDS.ETBF1, MHDS.ETBF2, and flag MHDS.FMBD is set. Updated only after the Host has reset flag MHDS.FMBD.
MBT	[22:16]	rh	**Message Buffer Transmitted** Number of last successfully transmitted Message Buffer. If the Message Buffer is configured for single-shot mode, the respective TXR flag in the Transmission Request Registers TXRQ1 to TXRQ4 was reset. MBT is reset when the Communication Controller leaves "CONFIG" state or enters "STARTUP" state.
MBU	[30:24]	rh	**Message Buffer Updated** Number of Message Buffer that was updated last. For this Message Buffer the respective NDn (n = 0-31) to NDn (n = 96-127)and / or MBCn (n = 0-31) to MBCn (n = 96-127) flag in the New Data Registers NDAT1 to NDAT4 and the Message Buffer Status Changed MBSC1 to MBSC4 registers are also set. MBU is reset when the Communication Controller leaves "CONFIG" state or enters "STARTUP" state.
0	15, 23, 31	r	**Reserved** Returns 0 if read; should be written with 0.

Last Dynamic Transmit Slot (LDTS)

The Last Dynamic Transmit Slot Register stores the Slot Counter value at the time of the last Frame transmission in the dynamic segment. This register is reset when the Communication Controller leaves “CONFIG” state or enters “STARTUP” state.

LDTS
Last Dynamic Transmit Slot **(0314_H)** **Reset Value: $0000\ 0000_H$**

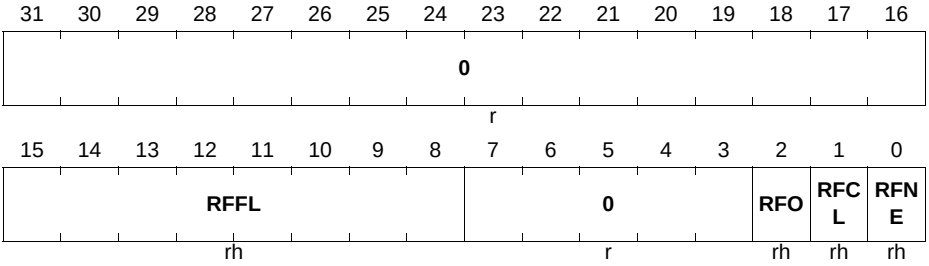


Field	Bits	Type	Description
LDTA	[10:0]	rh	**Last Dynamic Transmission Channel A** Value of (vSlotCounter[A]) at the time of the last Frame transmission on channel A in the dynamic segment of this node. It is updated at the end of the dynamic segment and is reset to zero if no Frame was transmitted during the dynamic segment.
LDTB	[26:16]	rh	**Last Dynamic Transmission Channel B** Value of (vSlotCounter[B]) at the time of the last Frame transmission on channel B in the dynamic segment of this node. It is updated at the end of the dynamic segment and is reset to zero if no Frame was transmitted during the dynamic segment.
0	[15:11], [31:27]	r	**Reserved** Returns 0 if read; should be written with 0.

FIFO Status Register (FSR)

The register is reset when the Communication Controller leaves “CONFIG” state or enters “STARTUP” state.

FSR
FIFO Status Register **(0318_H)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RFNE	0	rh	**Receive FIFO Not Empty** This flag is set by the Communication Controller when a received valid Frame (data or NULL Frame depending on rejection mask) was stored in the FIFO. In addition, service request flag SIR.RFNE is set. The bit is reset after the Host has read all message from the FIFO. 0_B Receive FIFO is empty 1_B Receive FIFO is not empty
RFCL	1	rh	**Receive FIFO Critical Level** This flag is set when the receive FIFO fill level RFFL is equal or greater than the critical level as configured by FCL.CL. The flag is cleared by the Communication Controller as soon as RFFL drops below FCL.CL. When RFCL changes from 0 to 1 bit SIR.RFCL is set to 1, and if enabled, an service request is generated. 0_B Receive FIFO below critical level 1_B Receive FIFO critical level reached

Field	Bits	Type	Description
RFO	2	rh	**Receive FIFO Overrun** The flag is set by the Communication Controller when a receive FIFO overrun is detected. When a receive FIFO overrun occurs, the oldest message is overwritten with the actual received message. In addition, service request flag EIR.RFO is set.The flag is cleared by the next FIFO read access issued by the Host. 0_B No receive FIFO overrun detected 1_B A receive FIFO overrun has been detected
RFFL	[15:8]	rh	**Receive FIFO Fill Level** Number of FIFO buffers filled up with new data not yet read by the Host. Maximum value is 128.
0	[7:3], [31:16]	r	**Reserved** Returns 0 if read; should be written with 0.

Message Handler Constraints Flags (MHDF)

Some constraints exist for the Message Handler regarding f_{CLC_ERAY} frequency, Message RAM configuration, and FlexRay™ bus traffic. To simplify software development, constraints violations are reported by setting flags in the MHDF. The register is reset when the Communication Controller leaves “CONFIG” state or enters “STARTUP” state. A flag is cleared by setting the corresponding bit position. Clearing has no effect on the flag. If any flag in MHDFL is set, interrupt flag EIR.MHF is set.

MHDF

Message Handler Constraints Flags (031C$_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							WAHP	TNSB	TNSA	TBFB	TBFA	FNFB	FNFA	SNUB	SNUA
r							rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
SNUA	0	rwh	**Status Not Updated Channel A** This flag is set by the Communication Controller when the Message Handler, due to overload condition, was not able to update a Message Buffer’s status MBS with respect to channel A. 0_B No overload condition occurred when updating MBS for channel A 1_B MBS for channel A not updated
SNUB	1	rwh	**Status Not Updated Channel B** This flag is set by the Communication Controller when the Message Handler, due to overload condition, was not able to update a Message Buffer’s status MBS with respect to channel B. 0_B No overload condition occurred when updating MBS for channel B 1_B MBS for channel B not updated

Field	Bits	Type	Description
FNFA	2	rwh	**Find Sequence Not Finished Channel A** This flag is set by the Communication Controller when the Message Handler, due to overload condition, was not able to finish a find sequence (scan of Message RAM for matching Message Buffer) with respect to channel A. 0_B No find sequence not finished for channel A 1_B Find sequence not finished for channel A
FNFB	3	rwh	**Find Sequence Not Finished Channel B** This flag is set by the Communication Controller when the Message Handler, due to overload condition, was not able to finish a find sequence (scan of Message RAM for matching Message Buffer) with respect to channel B. 0_B No find sequence not finished for channel B 1_B Find sequence not finished for channel B
TBFA	4	rwh	**Transient Buffer Access Failure A** This flag is set by the Communication Controller when a read or write access to Transient Buffer A requested by PRT A could not complete within the available time. 0_B No TBF A access failure 1_B TBF A access failure
TBFB	5	rwh	**Transient Buffer Access Failure B** This flag is set by the Communication Controller when a read or write access to Transient Buffer B requested by PRT B could not complete within the available time. 0_B No Transient Buffer B access failure 1_B Transient Buffer B access failure
TNSA	6	rwh	**Transmission Not Started Channel A** This flag is set by the CC when the Message Handler was not ready to start a scheduled transmission on channel A at the action point of the configured slot. 0_B No transmission not started on channel A 1_B Transmission not started on channel A
TNSB	7	rwh	**Transmission Not Started Channel B** This flag is set by the CC when the Message Handler was not ready to start a scheduled transmission on channel B at the action point of the configured slot. 0_B No transmission not started on channel B 1_B Transmission not started on channel B

Field	Bits	Type	Description
WAHP	8	rwh	**Write Attempt to Header Partition** Outside “DEFAULT_CONFIG” and “CONFIG” state this flag is set by the Communication Controller when the message handler tries to write message data into the Header Partition of the Message RAM due to faulty configuration of a Message Buffer. The write attempt is not executed, to protect the Header Partition from unintended write accesses. 0_B No write attempt to Header Partition 1_B Write attempt to Header Partition
0	[31:9]	r	**Reserved** Returns 0 if read; should be written with 0.

Transmission Request 1 (TXRQ1)

This register reflect the state of the TXR flags of the configured Message Buffers 0 to 31. The flags are evaluated for transmit buffers only. If the number of configured Message Buffers is less than 31, the remaining TXRn flags have no meaning and are read as 0.

TXRQ1
Transmission Request Register 1 (0320_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXR 31	**TXR 30**	**TXR 29**	**TXR 28**	**TXR 27**	**TXR 26**	**TXR 25**	**TXR 24**	**TXR 23**	**TXR 22**	**TXR 21**	**TXR 20**	**TXR 19**	**TXR 18**	**TXR 17**	**TXR 16**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXR 15	**TXR 14**	**TXR 13**	**TXR 12**	**TXR 11**	**TXR 10**	**TXR 9**	**TXR 8**	**TXR 7**	**TXR 6**	**TXR 5**	**TXR 4**	**TXR 3**	**TXR 2**	**TXR 1**	**TXR 0**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
TXRn (n = 0-31)	n	rh	**Transmission Request n (n = 0-31)** If the flag is set, the respective Message Buffer 0 to 31 is ready for transmission respectively transmission of this Message Buffer is in progress. In single-shot mode the flags are reset after transmission has completed.

Transmission Request Register 2 (TXRQ2)

This register reflect the state of the TXR flags of the configured Message Buffers 31 to 63. The flags are evaluated for transmit buffers only. If the number of configured Message Buffers is less than 63, the remaining TXRn flags have no meaning and are read as 0.

TXRQ2
Transmission Request Register 2 (0324_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXR 63	**TXR 62**	**TXR 61**	**TXR 60**	**TXR 59**	**TXR 58**	**TXR 57**	**TXR 56**	**TXR 55**	**TXR 54**	**TXR 53**	**TXR 52**	**TXR 51**	**TXR 50**	**TXR 49**	**TXR 48**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXR 47	**TXR 46**	**TXR 45**	**TXR 44**	**TXR 43**	**TXR 42**	**TXR 41**	**TXR 40**	**TXR 39**	**TXR 38**	**TXR 37**	**TXR 36**	**TXR 35**	**TXR 34**	**TXR 33**	**TXR 32**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
TXRn (n = 32-63)	n - 32	rh	**Transmission Request n (n = 32-63)** If the flag is set, the respective Message Buffer 32 to 63 is ready for transmission respectively transmission of this Message Buffer is in progress. In single-shot mode the flags are reset after transmission has completed.

Transmission Request Register 3 (TXRQ3)

This register reflect the state of the TXR flags of the configured Message Buffers 64 to 95. The flags are evaluated for transmit buffers only. If the number of configured Message Buffers is less than 95, the remaining TXRn flags have no meaning and are read as 0.

TXRQ3

Transmission Request Register 3 (0328_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXR 95	**TXR 94**	**TXR 93**	**TXR 92**	**TXR 91**	**TXR 90**	**TXR 89**	**TXR 88**	**TXR 87**	**TXR 86**	**TXR 85**	**TXR 84**	**TXR 83**	**TXR 82**	**TXR 81**	**TXR 80**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXR 79	**TXR 78**	**TXR 77**	**TXR 76**	**TXR 75**	**TXR 74**	**TXR 73**	**TXR 72**	**TXR 71**	**TXR 70**	**TXR 69**	**TXR 68**	**TXR 67**	**TXR 66**	**TXR 65**	**TXR 64**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
TXRn (n = 64-95)	n - 64	rh	**Transmission Request n (n = 64-95)** If the flag is set, the respective Message Buffer 64 to 95 is ready for transmission respectively transmission of this Message Buffer is in progress. In single-shot mode the flags are reset after transmission has completed.

Transmission Request Register 4 (TXRQ4)

This register reflect the state of the TXR flags of the configured Message Buffers 96 to 127. The flags are evaluated for transmit buffers only. If the number of configured Message Buffers is less than 127, the remaining TXRn flags have no meaning and are read as 0.

TXRQ4
Transmission Request Register 4 ($032C_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXR 127	**TXR 126**	**TXR 125**	**TXR 124**	**TXR 123**	**TXR 122**	**TXR 121**	**TXR 120**	**TXR 119**	**TXR 118**	**TXR 117**	**TXR 116**	**TXR 115**	**TXR 114**	**TXR 113**	**TXR 112**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXR 111	**TXR 110**	**TXR 109**	**TXR 108**	**TXR 107**	**TXR 106**	**TXR 105**	**TXR 104**	**TXR 103**	**TXR 102**	**TXR 101**	**TXR 100**	**TXR 99**	**TXR 98**	**TXR 97**	**TXR 96**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
TXRn (n = 96-127)	n - 96	rh	**Transmission Request n (n = 96-127)** If the flag is set, the respective Message Buffer 96 to 127 is ready for transmission respectively transmission of this Message Buffer is in progress. In single-shot mode the flags are reset after transmission has completed.

New Data Register 1 (NDAT1)

This register reflect the state of the ND flags of all configured Message Buffers 0 to 31. ND flags assigned to transmit buffers are meaningless. If the number of configured Message Buffers is less than 31, the remaining NDn flags have no meaning. The registers are reset when the Communication Controller leaves “CONFIG” state or enters “STARTUP” state.

NDAT1
New Data Register 1 **(0330_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ND 31	**ND 30**	**ND 29**	**ND 28**	**ND 27**	**ND 26**	**ND 25**	**ND 24**	**ND 23**	**ND 22**	**ND 21**	**ND 20**	**ND 19**	**ND 18**	**ND 17**	**ND 16**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ND 15	**ND 14**	**ND 13**	**ND 12**	**ND 11**	**ND 10**	**ND9**	**ND8**	**ND7**	**ND6**	**ND5**	**ND4**	**ND3**	**ND2**	**ND1**	**ND0**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
NDn (n = 0-31)	n	rh	**New Data n (n = 0-31)** The flags are set when a valid received Data Frame matches the Message Buffer’s filter configuration, independent of the payload length received or the payload length configured for that Message Buffer. The flags are not set after reception of NULL Frames except for Message Buffers belonging to the receive FIFO. An ND flag is reset when the Header Section of the corresponding Message Buffer is reconfigured or when the Data Section has been transferred to the Output Buffer.

New Data Register 2 (NDAT2)

This register reflect the state of the ND flags of all configured Message Buffers 32 to 63. ND flags assigned to transmit buffers are meaningless. If the number of configured Message Buffers is less than 63, the remaining NDn flags have no meaning. The registers are reset when the Communication Controller leaves “CONFIG” state or enters “STARTUP” state.

NDAT2
New Data Register 2 **(0334_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ND 63	**ND 62**	**ND 61**	**ND 60**	**ND 59**	**ND 58**	**ND 57**	**ND 56**	**ND 55**	**ND 54**	**ND 53**	**ND 52**	**ND 51**	**ND 50**	**ND 49**	**ND 48**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ND 47	**ND 46**	**ND 45**	**ND 44**	**ND 43**	**ND 42**	**ND 41**	**ND 40**	**ND 39**	**ND 38**	**ND 37**	**ND 36**	**ND 35**	**ND 34**	**ND 33**	**ND 32**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
NDn (n = 32-63)	n - 32	rh	**New Data n (n = 32-63)** The flags are set when a valid received Data Frame matches the Message Buffer’s filter configuration, independent of the payload length received or the payload length configured for that Message Buffer. The flags are not set after reception of NULL Frames except for Message Buffers belonging to the receive FIFO. An ND flag is reset when the Header Section of the corresponding Message Buffer is reconfigured or when the Data Section has been transferred to the Output Buffer.

New Data Register 3 (NDAT3)

This register reflect the state of the ND flags of all configured Message Buffers 64 to 95. ND flags assigned to transmit buffers are meaningless. If the number of configured Message Buffers is less than 95, the remaining NDn flags have no meaning. The registers are reset when the Communication Controller leaves “CONFIG” state or enters “STARTUP” state.

NDAT3
New Data Register 3 **(0338_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ND 95	**ND 94**	**ND 93**	**ND 92**	**ND 91**	**ND 90**	**ND 89**	**ND 88**	**ND 87**	**ND 86**	**ND 85**	**ND 84**	**ND 83**	**ND 82**	**ND 81**	**ND 80**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ND 79	**ND 78**	**ND 77**	**ND 76**	**ND 75**	**ND 74**	**ND 73**	**ND 72**	**ND 71**	**ND 70**	**ND 69**	**ND 68**	**ND 67**	**ND 66**	**ND 65**	**ND 64**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
NDn (n = 64-95)	n - 64	rh	**New Data n (n = 64-95)** The flags are set when a valid received Data Frame matches the Message Buffer’s filter configuration, independent of the payload length received or the payload length configured for that Message Buffer. The flags are not set after reception of NULL Frames except for Message Buffers belonging to the receive FIFO. An ND flag is reset when the Header Section of the corresponding Message Buffer is reconfigured or when the Data Section has been transferred to the Output Buffer.

New Data Register 4 (NDAT4)

This register reflect the state of the ND flags of all configured Message Buffers 96 to 127. ND flags assigned to transmit buffers are meaningless. If the number of configured Message Buffers is less than 127, the remaining NDn flags have no meaning. The registers are reset when the Communication Controller leaves “CONFIG” state or enters “STARTUP” state.

NDAT4
New Data Register 4 **($033C_H$)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ND 127	**ND 126**	**ND 125**	**ND 124**	**ND 123**	**ND 122**	**ND 121**	**ND 120**	**ND 119**	**ND 118**	**ND 117**	**ND 116**	**ND 115**	**ND 114**	**ND 113**	**ND 112**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ND 111	**ND 110**	**ND 109**	**ND 108**	**ND 107**	**ND 106**	**ND 105**	**ND 104**	**ND 103**	**ND 102**	**ND 101**	**ND 100**	**ND 99**	**ND 98**	**ND 97**	**ND 96**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
NDn (n = 96-127)	n - 96	rh	**New Data n (n = 96-127)** The flags are set when a valid received Data Frame matches the Message Buffer’s filter configuration, independent of the payload length received or the payload length configured for that Message Buffer. The flags are not set after reception of NULL Frames except for Message Buffers belonging to the receive FIFO. An ND flag is reset when the Header Section of the corresponding Message Buffer is reconfigured or when the Data Section has been transferred to the Output Buffer.

Message Buffer Status Changed 1 (MBSC1)

This register reflect the state of the MBC flags of all configured Message Buffers. If the number of configured Message Buffers is less than 31, the remaining MBCn flags have no meaning. The register is reset when the communication controller leaves “CONFIG” state or enters “STARTUP” state.

MBSC1
Message Buffer Status Changed 1 (0340_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MBC 31	**MBC 30**	**MBC 29**	**MBC 28**	**MBC 27**	**MBC 26**	**MBC 25**	**MBC 24**	**MBC 23**	**MBC 22**	**MBC 21**	**MBC 20**	**MBC 19**	**MBC 18**	**MBC 17**	**MBC 16**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MBC 15	**MBC 14**	**MBC 13**	**MBC 12**	**MBC 11**	**MBC 10**	**MBC 9**	**MBC 8**	**MBC 7**	**MBC 6**	**MBC 5**	**MBC 4**	**MBC 3**	**MBC 2**	**MBC 1**	**MBC 0**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
MBCn (n = 0-31)	n	rh	**Message Buffer Status Changed n (n = 0-31)** An MBC flags is set whenever the Message Handler changes on of the status flags VFRA, VFRB, SEOA, SEOB, CEOA, CEOB, SVOA, SVOB, TCIA, TCIB, ESA, ESB, MLST, FTA, FTB in the Header Section (see **“Message Buffer Status (MBS)” on Page 23-176**) of the respective Message Buffer 0 to Message Buffer 31. The flags are reset when the Header Section of the Message Buffer is reconfigured or when it has been transferred to the Output Buffer.

Message Buffer Status Changed 2 (MBSC2)

This register reflect the state of the MBC flags of all configured Message Buffers. If the number of configured Message Buffers is less than 63, the remaining MBCn flags have no meaning. The register is reset when the communication controller leaves “CONFIG” state or enters “STARTUP” state.

MBSC2

Message Buffer Status Changed 2 (0344_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MBC 63	**MBC 62**	**MBC 61**	**MBC 60**	**MBC 59**	**MBC 58**	**MBC 57**	**MBC 56**	**MBC 55**	**MBC 54**	**MBC 53**	**MBC 52**	**MBC 51**	**MBC 50**	**MBC 49**	**MBC 48**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MBC 47	**MBC 46**	**MBC 45**	**MBC 44**	**MBC 43**	**MBC 42**	**MBC 41**	**MBC 40**	**MBC 39**	**MBC 38**	**MBC 37**	**MBC 36**	**MBC 35**	**MBC 34**	**MBC 33**	**MBC 32**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
MBCn (n = 32-63)	n - 32	rh	**Message Buffer Status Changed n (n = 32-63)** An MBC flags is set whenever the Message Handler changes on of the status flags VFRA, VFRB, SEOA, SEOB, CEOA, CEOB, SVOA, SVOB, TCIA, TCIB, ESA, ESB, MLST, FTA, FTB in the Header Section (see **“Message Buffer Status (MBS)” on Page 23-176**) of the respective Message Buffer 32 to Message Buffer 63. The flags are reset when the Header Section of the Message Buffer is reconfigured or when it has been transferred to the Output Buffer.

Message Buffer Status Changed 3 (MBSC3)

This register reflect the state of the MBC flags of all configured Message Buffers. If the number of configured Message Buffers is less than 95, the remaining MBCn flags have no meaning. The register is reset when the communication controller leaves “CONFIG” state or enters “STARTUP” state.

MBSC3

Message Buffer Status Changed 3 (0348_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MBC 95	**MBC 94**	**MBC 93**	**MBC 92**	**MBC 91**	**MBC 90**	**MBC 89**	**MBC 88**	**MBC 87**	**MBC 86**	**MBC 85**	**MBC 84**	**MBC 83**	**MBC 82**	**MBC 81**	**MBC 80**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MBC 79	**MBC 78**	**MBC 77**	**MBC 76**	**MBC 75**	**MBC 74**	**MBC 73**	**MBC 72**	**MBC 71**	**MBC 70**	**MBC 69**	**MBC 68**	**MBC 67**	**MBC 66**	**MBC 65**	**MBC 64**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
MBCn (n = 64-95)	n - 64	rh	**Message Buffer Status Changed n (n = 64-95)** An MBC flags is set whenever the Message Handler changes on of the status flags VFRA, VFRB, SEOA, SEOB, CEOA, CEOB, SVOA, SVOB, TCIA, TCIB, ESA, ESB, MLST, FTA, FTB in the Header Section (see **“Message Buffer Status (MBS)” on Page 23-176**) of the respective Message Buffer 64 to Message Buffer 95. The flags are reset when the Header Section of the Message Buffer is reconfigured or when it has been transferred to the Output Buffer.

Message Buffer Status Changed 4 (MBSC4)

This register reflect the state of the MBC flags of all configured Message Buffers. If the number of configured Message Buffers is less than 127, the remaining MBCn flags have no meaning. The register is reset when the communication controller leaves “CONFIG” state or enters “STARTUP” state.

MBSC4

Message Buffer Status Changed 4 ($034C_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MBC 127	**MBC 126**	**MBC 125**	**MBC 124**	**MBC 123**	**MBC 122**	**MBC 121**	**MBC 120**	**MBC 119**	**MBC 118**	**MBC 117**	**MBC 116**	**MBC 115**	**MBC 114**	**MBC 113**	**MBC 112**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MBC 111	**MBC 110**	**MBC 109**	**MBC 108**	**MBC 107**	**MBC 106**	**MBC 105**	**MBC 104**	**MBC 103**	**MBC 102**	**MBC 101**	**MBC 100**	**MBC 99**	**MBC 98**	**MBC 97**	**MBC 96**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
MBCn (n = 96-127)	n - 96	rh	**Message Buffer Status Changed n (n = 96-127)** An MBC flags is set whenever the Message Handler changes on of the status flags VFRA, VFRB, SEOA, SEOB, CEOA, CEOB, SVOA, SVOB, TCIA, TCIB, ESA, ESB, MLST, FTA, FTB in the Header Section (see **“Message Buffer Status (MBS)” on Page 23-176**) of the respective Message Buffer 96 to Message Buffer 127. The flags are reset when the Header Section of the Message Buffer is reconfigured or when it has been transferred to the Output Buffer.

23.5.2.8 Identification Registers

Core Release Register (CREL)

This register contains bit fields about the ERAY module identification. It is read only.

CREL
Core Release Register **($03F0_H$)** **Reset Value: $XXXX\ XXXX_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REL				STEP				SUB STEP				YEAR				MON								DAY							
r				r				r				r				r								r							

Field	Bits	Type	Description
DAY	[7:0]	r	**Design Time Stamp, Day** Two digits, BCD-coded.
MON	[15:8]	r	**Design Time Stamp, Month** Two digits, BCD-coded.
YEAR	[19:16]	r	**Design Time Stamp, Year** One digit, BCD-coded.
SUBSTEP	[23:20]	r	**Sub-Step of Core Release** One digits, BCD-coded. 0_H Alpha, pre-Beta, pre-Beta-update, pre-Beta2, pre-Beta2-update, Beta, Beta2, Revision 1.0.0 1_H Beta_ct, Beta-ct-fix1, Revision 1.0.1 2_H Revision1.0RC1,Beta-ct-fix2, REVISION 1.0RC1
STEP	[27:24]	r	**Step of Core Release** One digits, BCD-coded. 0_H Revision 1.0.0 1_H Alpha 2_H pre-Beta 3_H pre-Beta-update 4_H pre-Beta2 5_H pre-Beta2-update 6_H Beta 7_H Beta2

Field	Bits	Type	Description
REL	[31:28]	r	**Core Release** One digit, BCD-coded. 0_B alpha…beta2ct 1_B Revision 1.0

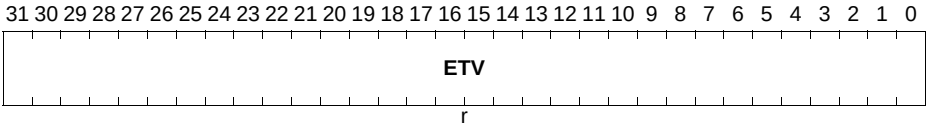
Table 23-7 Coding of releases

Release	Step	Sub-Step	Name	Release Date
0	1	0	Alpha	
0	2	0	pre-Beta	
0	3	0	pre-Beta-update	
0	4	0	pre-Beta2	
0	5	0	pre-Beta2-update	
0	6	0	Beta	
0	6	1	Beta-ct-fix1	14.10.2005
0	6	2	Beta-ct-fix2	14.12.2005
0	7	0	Beta2	03.02.2006
0	7	1	Beta2ct	24.03.2006
0	7	2	Revision 1.0RC1	07.04.2006
1	0	0	Release 1.0.0	19.05.2006
1	0	1	Release 1.0.1	2006
1	0	2	Release 1.0.2	31.10.2007

Endian Register (ENDN)

This register may be used to check, if the data of the E-Ray is handled by a host with the correct endian format. It is read only.

ENDN
Endian Register **($003F4_H$)** **Reset Value: $8765\ 4321_H$**



Field	Bits	Type	Description
ETV	[31:0]	r	**Endianness Test Value** The endianness test value.

23.5.2.9 Input Buffer

Double buffer structure consisting of Input Buffer Host and Input Buffer Shadow. While the Host can write to Input Buffer Host, the transfer to the Message RAM is done from Input Buffer Shadow. The Input Buffer holds the Header and Data Sections to be transferred to the selected Message Buffer in the Message RAM. It is used to configure the Message Buffers in the Message RAM and to update the Data Sections of transmit buffers.

When updating the Header Section of a Message Buffer in the Message RAM from the Input Buffer, the Message Buffer Status as described in **“Message Buffer Status (MBS)” on Page 23-176** is automatically reset to zero.

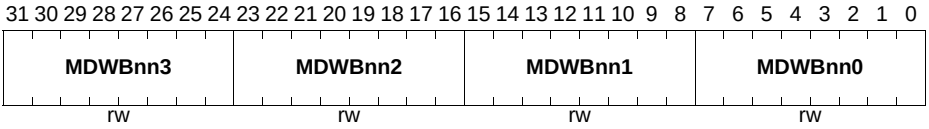
The Header Sections of Message Buffers belonging to the receive FIFO can only be (re)configured when the Communication Controller is in “DEFAULT_CONFIG” or “CONFIG” state. For those Message Buffers only the payload length configured and the data pointer need to be configured via WRHS2.PLC and WRHS2.DP. All information required for acceptance filtering is taken from the FIFO rejection filter and the FIFO rejection filter mask.

The data transfer between Input Buffer (IBF) and Message RAM is described in detail in **“Data Transfer from Input Buffer to Message RAM” on Page 23-224**.

Write Data Section [01 - 64] (WRDSnn (nn = 01-64))

The Write Data Section (WRDSnn, nn = 01-64) holds the data words to be transferred to the Data Section of the addressed Message Buffer. The data words (DW_n) are written to the Message RAM in transmission order from DW1 (byte0, byte1) to DW_{PL} (PL = number of data words as defined by the payload length configured by WRHS2.PLC).

WRDSnn (nn = 01-64)
Write Data Section nn **($03FC_H$ + nn * 4)** **Reset Value: $0000\ 0000_H$**



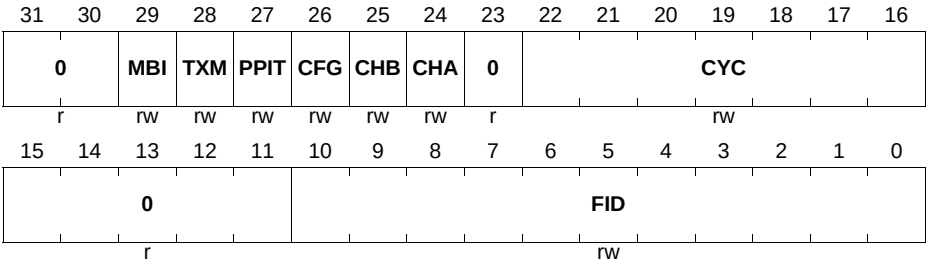
Field	Bits	Type	Description
MDWB0	[7:0]	rw	**32-Bit Word nn, Byte 0**
MDWB1	[15:8]	rw	**32-Bit Word nn, Byte 1**
MDWB2	[23:16]	rw	**32-Bit Word nn, Byte 2**
MDWB3	[31:24]	rw	**32-Bit Word nn, Byte 3**

Note: 16-bit Word 127 is located on WRDS64.MDW. In this case WRDS64.MDW is unused (no valid data). The Input Buffer RAMs are initialized to zero when leaving application reset or by CHI command CLEAR_RAMS.

Note: When writing to the WRDSnn (nn = 01-64), each 32-bit word has to be filled up by one 32-bit access OR two consecutive 16-bit accesses OR four consecutive 8-bit accesses before the transfer from the Input Buffer to the Message RAM is started by writing the number of the target Message Buffer in the Message RAM to the Input Buffer Command Request register. If a 32-bit word of the Input Buffer has been filled with less then two consecutive 16-bit accesses OR four consecutive 8-bit accesses (less then 32-bit), random data is transferred into the Input buffer for every not written 16-bit or 8-bit of a 32-bit word.

Write Header Section 1 (WRHS1)

WRHS1
Write Header Section 1 **(0500_H)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
FID	[10:0]	rw	**Frame ID** Frame ID of the selected Message Buffer. The Frame ID defines the slot number for transmission / reception of the respective message. Message Buffers with Frame ID = 0 are considered as not valid.
CYC	[22:16]	rw	**Cycle Code** The 7-bit cycle code determines the cycle set used for cycle counter filtering. For details about the configuration of the cycle code see **Section 23.6.7.3**.
CHA	24	rw	**Channel Filter Control A** The channel filtering field A associated with the buffer serves of channel A as a filter for receive buffers, and as a control field for transmit buffers
CHB	25	rw	**Channel Filter Control B** The channel filtering field B associated with the buffer serves of channel B as a filter for receive buffers, and as a control field for transmit buffers
CFG	26	rw	**Message Buffer Direction Configuration Bit** This bit is used to configure the corresponding buffer as a transmit buffer or as a receive buffer. For Message Buffers belonging to the receive FIFO the bit is not evaluated. 0_B The corresponding buffer is configured as Receive Buffer 1_B The corresponding buffer is configured as Transmit Buffer

Field	Bits	Type	Description
PPIT	27	rw	**Payload Preamble Indicator Transmit** This bit is used to control the state of the Payload Preamble Indicator in transmit Frames. If the bit is set in a static Message Buffer, the respective Message Buffer holds Network Management information. If the bit is set in a dynamic Message Buffer the first two byte of the Payload Segment may be used for message ID filtering by the receiver. Message ID filtering of received FlexRay™ Frames is not supported by the E-Ray module, but can be done by the Host. 0_B Payload Preamble Indicator not set 1_B Payload Preamble Indicator set
TXM	28	rw	**Transmission Mode** This bit is used to select the transmission mode (see **“Transmit Buffers” on Page 23-217**). 0_B Continuous mode 1_B Single-shot mode
MBI	29	rw	**Message Buffer Service Request** This bit enables the receive / transmit service request for the corresponding Message Buffer. After a dedicated receive buffer has been updated by the Message Handler, flag SIR.RXI and /or SIR.MBSI in the Status Service Request register are set. After a transmission has completed flag SIR.TXI is set. 0_B The corresponding Message Buffer service request is disabled 1_B The corresponding Message Buffer service request is enabled
0	[15:11], 23, [31:30]	r	**Reserved** Returns 0 if read; should be written with 0.

Note: The Input Buffer RAMs are initialized to zero when leaving application reset or by CHI command CLEAR_RAMS. Note that only the currently active IBF bank is cleared. To clear the 2nd bank as well, CUST1.IBF1PAG and CUST1.IBF2PAG need to be set and command CLEAR_RAMS need to be issued again. This is required in particular after an application reset. If the 2nd bank of IBF is left unused, this procedure is not required.

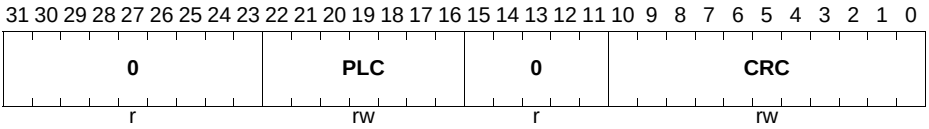
Table 23-8 Channel Filter Control Bits

CHA	CHB	Transmit Buffer transmit Frame on	Receive Buffer store Frame received from
1[1)]	1[1)]	Both Channels (static segment only)	Channel A or B (store first semantically valid Frame, static segment only)
1	0	Channel A	Channel A
0	1	Channel B	Channel B
0	0	No Transmission	Ignore Frame

1) If a Message Buffer is configured for the dynamic segment and both bits of the channel filtering field are set to 1, no Frames are transmitted resp. received Frames are ignored (same function as CHA = CHB = 0)

Write Header Section 2 (WRHS2)

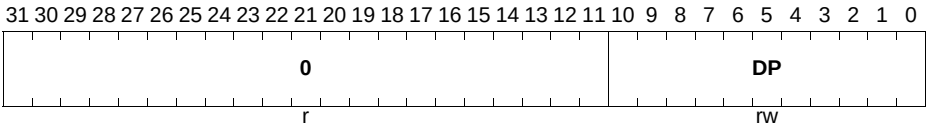
WRHS2
Write Header Section 2 **(0504_H)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CRC	[10:0]	rw	**Header CRC** (vRF!Header!HeaderCRC) Receive Buffer: Configuration not required Transmit Buffer: Header CRC calculated and configured by the Host. For calculation of the Header CRC the payload length of the Frame send on the bus has to be considered. In static segment the payload length of all Frames is configured by MHDC.SFDL.
PLC	[22:16]	rw	**Payload Length Configured** Length of Data Section (number of 2-byte words) as configured by the Host. During static segment the static Frame payload length as configured by MHDC.SFDL in the MHD Configuration Register defines the payload length for all static Frames. If the payload length configured by PLC is shorter than this value padding byte are inserted to ensure that Frames have proper physical length. The padding pattern is logical zero.
0	[15:11], [31:23]	r	**Reserved** Returns 0 if read; should be written with 0.

Write Header Section 3 (WRHS3)

WRHS3
Write Header Section 3 **(0508_H)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
DP	[10:0]	rw	**Data Pointer** Pointer to the first 32-bit word of the Data Section of the addressed Message Buffer in the Message RAM.
0	[31:11]	r	**Reserved** Returns 0 if read; should be written with 0.

Input Buffer Command Mask (IBCM)

Configures how the Message Buffer in the Message RAM selected by the Input Buffer Command Request register IBCR is updated. If IBF Host and IBF Shadow are swapped, also masked bits IBCM.LHSH, IBCM.LDSH, and IBCM.STXRH are swapped with bits IBCM.LHSS, IBCM.LDSS, and IBCM.STXRS to keep them attached to the respective Input Buffer transfer.

IBCM
Input Buffer Command Mask **(0510_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0													STXRS	LDSS	LHSS
r													rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													STXRH	LDSH	LHSH
r													rwh	rwh	rwh

Field	Bits	Type	Description
LHSH	0	rwh	**Load Header Section Host** 0_B Header Section is not updated 1_B Header Section selected for transfer from Input Buffer to the Message RAM
LDSH	1	rwh	**Load Data Section Host** 0_B Data Section is not updated 1_B Data Section selected for transfer from Input Buffer to the Message RAM
STXRH	2	rwh	**Set Transmission Request Host** If this bit is set to 1, the Transmission Request flag TXRQ1.TXRn (n = 0-31) to TXRQ4.TXRn (n = 0-31) for the selected Message Buffer is set in the Transmission Request Registers to release the Message Buffer for transmission. In single-shot mode the flag is cleared by the Communication Controller after transmission has completed. TXRQ1.TXRn (n = 0-31) to TXRQ4.TXRn (n = 0-31) are evaluated for transmit buffer only. 0_B Reset Transmission Request flag 1_B Set Transmission Request flag, transmit buffer released for transmission

Field	Bits	Type	Description
LHSS	16	rh	**Load Header Section Shadow** 0_B Header Section is not updated 1_B Header Section selected for transfer from Input Buffer to the Message RAM (transfer is ongoing of finalized)
LDSS	17	rh	**Load Data Section Shadow** 0_B Data Section is not updated 1_B Data Section selected for transfer from Input Buffer to the Message RAM (transfer is ongoing of finalized)
STXRS	18	rh	**Transmission Request Shadow** If this bit is set to 1, the Transmission Request flag TXRQ1.TXRn (n = 0-31) to TXRQ4.TXRn (n = 0-31) for the selected Message Buffer is set in the Transmission Request Registers to release the Message Buffer for transmission. In single-shot mode the flag is cleared by the Communication Controller after transmission has completed. TXRQ1.TXRn (n = 0-31) to TXRQ4.TXRn (n = 0-31) are evaluated for transmit buffer only. 0_B Reset Transmission Request flag 1_B Set Transmission Request flag, transmit buffer released for transmission (operation is ongoing of finalized)
0	[15:3], [31:19]	r	**Reserved** Returns 0 if read; should be written with 0.

Input Buffer Command Request (IBCR)

When the Host writes the number of the target Message Buffer in the Message RAM to IBRH in the Input Buffer Command Request register, IBF Host and IBF Shadow are swapped. In addition the Message Buffer numbers stored under IBRH and IBRS are also swapped (see also **“Data Transfer from Input Buffer to Message RAM” on Page 23-224**).

With this write operation the IBSYS bit in the Input Buffer Command Request register is set to 1. The Message Handler then starts to transfer the contents of IBF Shadow to the Message Buffer in the Message RAM selected by IBRS.

While the Message Handler transfers the data from IBF Shadow to the target Message Buffer in the Message RAM, the Host may write the next message into the IBF Host. After the transfer between IBF Shadow and the Message RAM has completed, the IBSYS bit is set back to 0 and the next transfer to the Message RAM may be started by the Host by writing the respective target Message Buffer number to IBRH.

If a write access to IBRH occurs while IBSYS is 1, IBSYH is set to 1. After completion of the ongoing data transfer from IBF Shadow to the Message RAM, IBF Host and IBF Shadow are swapped, IBSYH is reset to 0. IBSYS remains set to 1, and the next transfer to the Message RAM is started. In addition the Message Buffer numbers stored under IBRH and IBRS are also swapped. Any write access to an Input Buffer register while both IBSYS and IBSYH are set will cause the error flag EIR.IIBA to be set. In this case the Input Buffer will not be changed.

IBCR
Input Buffer Command Request (0514_H) Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IBSYS	0								IBRS						
rh	r								rh						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBSYH	0								IBRH						
rh	r								rwh						

Field	Bits	Type	Description
IBRH	[6:0]	rwh	**Input Buffer Request Host** Selects the target Message Buffer in the Message RAM for data transfer from Input Buffer. Valid values are 00_H to $7F_H$ (0…127).

Field	Bits	Type	Description
IBSYH	15	rh	**Input Buffer Busy Host** Set to 1 by writing IBRH while IBSYS is still 1. After the ongoing transfer between IBF Shadow and the Message RAM has completed, the IBSYH is set back to 0. 0_B No request pending 1_B Request while transfer between IBF Shadow and Message RAM in progress
IBRS	[22:16]	rh	**Input Buffer Request Shadow** Number of the target Message Buffer actually updated/lately updated. Valid values are 00_H to $7F_H$ (0…127).
IBSYS	31	rh	**Input Buffer Busy Shadow** Set to 1 after writing IBRH. When the transfer between IBF Shadow and the Message RAM has completed, IBSYS is set back to 0. 0_B Transfer between IBF Shadow and Message RAM completed 1_B Transfer between IBF Shadow and Message RAM in progress
0	[14:7], [30:23]	r	**Reserved** Returns 0 if read; should be written with 0.

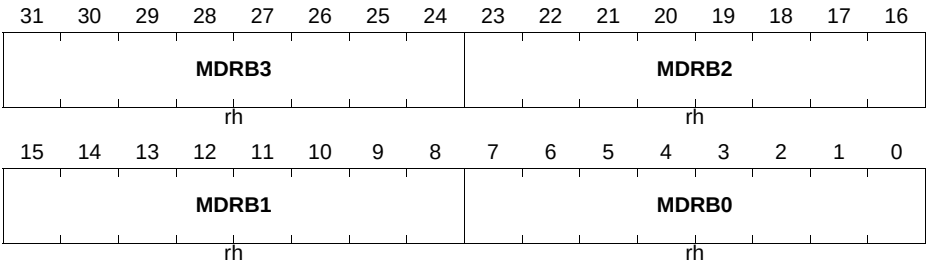
23.5.2.10 Output Buffer

Double buffer structure consisting of Output Buffer Host and Output Buffer Shadow. Used to read out Message Buffers from the Message RAM. While the Host can read from Output Buffer Host, the Message Handler transfers the selected Message Buffer from Message RAM to the respective Output Buffer Shadow. The data transfer between Message RAM and Output Buffer (OBF) is described in **"Data Transfer from Message RAM to Output Buffer" on Page 23-226**.

Read Data Section [1…64] (RDDSn)

The Read Data Section nn (RDDSnn, nn = 01-64) holds the data words read from the Data Section of the addressed Message Buffer. The data words are read from the Message RAM in reception order from DW1 (byte0, byte1) to DW_{PL} (PL = number of data words as defined by the Payload Length).

RDDSnn (nn = 01-64)
Read Data Section nn **($05FC_H$ + nn * 4)** **Reset Value: $0000\ 0000_H$**



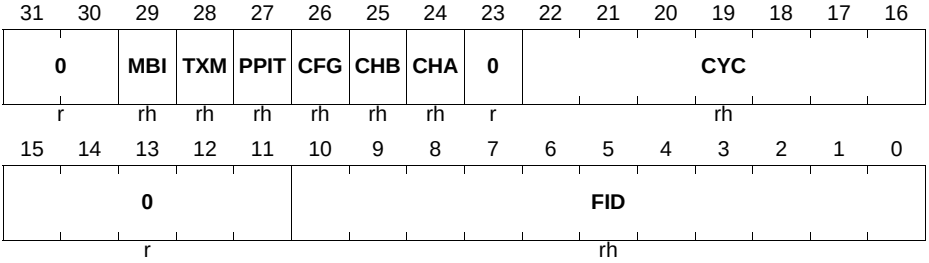
Field	Bits	Type	Description
MDRB0	[7:0]	rh	**32-Bit Word nn, Byte 0**
MDRB1	[15:8]	rh	**32-Bit Word nn, Byte 1**
MDRB2	[23:16]	rh	**32-Bit Word nn, Byte 2**
MDRB3	[31:24]	rh	**32-Bit Word nn, Byte 3**

Note: DW127 is located on RDDS64.MDW. In this case RDDS64.MDW is unused (no valid data). The Output Buffer RAMs are initialized to zero when leaving application reset or by CHI command CLEAR_RAMS.

Read Header Section 1 (RDHS1)

Values as configured by the Host via WRHS1 Register:

RDHS1
Read Header Section 1 **(0700_H)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
FID	[10:0]	rh	**Frame ID**
CYC	[22:16]	rh	**Cycle Code**
CHA	24	rh	**Channel Filter Control A**
CHB	25	rh	**Channel Filter Control B**
CFG	26	rh	**Message Buffer Direction Configuration Bit**
PPIT	27	rh	**Payload Preamble Indicator Transmit**
TXM	28	rh	**Transmission Mode**
MBI	29	rh	**Message Buffer Service Request**
0	[15:11], 23, [31:30]	r	**Reserved** Returns 0 if read; should be written with 0.

Note: In case that the Message Buffer read from the Message RAM belongs to the receive FIFO, FID holds the received Frame ID, while CYC, CHA, CHB, CFG, PPIT, TXM, and MBI are reset to zero.

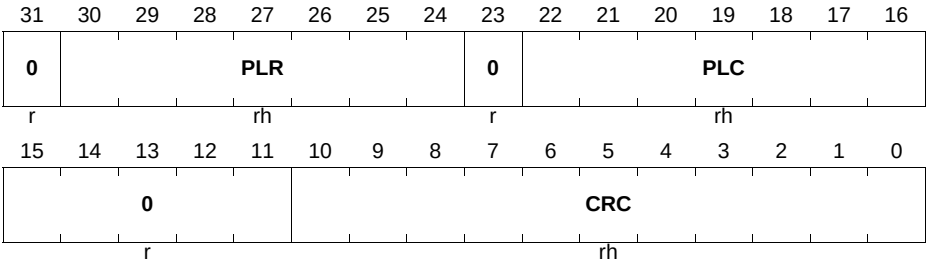
Table 23-9 Channel Filter Control Bits

CHA	CHB	Transmit Buffer transmit Frame on	Receive Buffer store Frame received from
1[1)]	1[1)]	Both Channels (static segment only)	Channel A or B (store first semantically valid Frame, static segment only)
1	0	Channel A	Channel A
0	1	Channel B	Channel B
0	0	No Transmission	Ignore Frame

1) If a Message Buffer is configured for the dynamic segment and both bits of the channel filtering field are set to 1, no Frames are transmitted resp. received Frames are ignored (same function as CHA = CHB = 0)

Read Header Section 2 (RDHS2)

RDHS2
Read Header Section 2 **(0704_H)** **Reset Value: $0000\ 0000_H$**



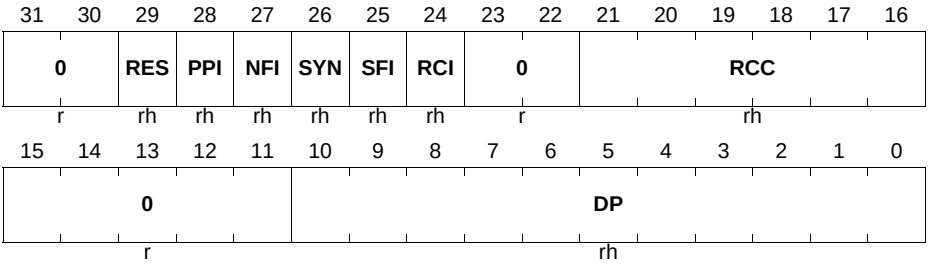
Field	Bits	Type	Description
CRC	[10:0]	rh	**Header CRC** (vRF!Header!HeaderCRC) Receive Buffer: Configuration not required. Header CRC updated from receive Data Frames. Transmit Buffer: Header CRC calculated and configured by the Host
PLC	[22:16]	rh	**Payload Length Configured** Length of Data Section (number of 2-byte words) as configured by the Host.

Field	Bits	Type	Description
PLR	[30:24]	rh	**Payload Length Received** (vRF!Header!Length) Payload length value updated from received Data Frame (exception: if Message Buffer belongs to the receive FIFO PLR is also updated from received NULL Frames). When a message is stored into a Message Buffer the following behavior with respect to payload length received and payload length configured is implemented: • **PLR > PLC:** The payload data stored in the Message Buffer is truncated to the payload length configured for even PLC or else truncated to PLC + 1. • **PLR ≤ PLC:** The received payload data is stored into the Message Buffers Data Section. The remaining data bytes of the Data Section as configured by PLC are filled with undefined data. • **PLR = 0:** The Message Buffer's Data Section is filled with undefined data. • **PLC = 0:** Message Buffer has no Data Section configured. No data is stored into the Message Buffer's Data Section.
0	[15:11], 23, 31	r	**Reserved** Returns 0 if read; should be written with 0.

Note: The Message RAM is organized in 4-byte words. When received data is stored into a Message Buffer's Data Section, the number of 2-byte data words written into the Message Buffer is PLC rounded to the next even value. PLC should be configured identical for all Message Buffers belonging to the receive FIFO. Header 2 is updated from Data Frames only.

Read Header Section 3 (RDHS3)

RDHS3
Read Header Section 3 **(0708_H)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
DP	[10:0]	rh	**Data Pointer** Pointer to the first 32-bit word of the Data Section of the addressed Message Buffer in the Message RAM.
RCC	[21:16]	rh	**Receive Cycle Count** (vRF!Header!CycleCount) Cycle counter value updated from received Data Frame.
RCI	24	rh	**Received on Channel Indicator** (vSS!Channel) Indicates the channel from which the received Data Frame was taken to update the respective receive buffer. 0_B Frame received on channel B 1_B Frame received on channel A
SFI	25	rh	**Startup Frame Indicator** (vRF!Header!SuFIndicator) A Startup Frame is marked by the Startup Frame indicator. 0_B The received Frame is not a startup Frame 1_B The received Frame is a startup Frame
SYN	26	rh	**SYNC Frame Indicator** (vRF!Header!SyFIndicator) A SYNC Frame is marked by the SYNC Frame indicator. 0_B The received Frame is not a SYNC Frame 1_B The received Frame is a SYNC Frame
NFI	27	rh	**NULL Frame Indicator** (vRF!Header!NFIndicator) Is set to 1 after storage of the first received Data Frame. 0_B Up to now no Data Frame has been stored into the respective Message Buffer 1_B At least one Data Frame has been stored into the respective Message Buffer

Field	Bits	Type	Description
PPI	28	rh	**Payload Preamble Indicator** (vRF!Header!PPIndicator) The payload preamble indicator defines whether a Network Management vector or message ID is contained within the Payload Segment of the received Frame. 0_B The Payload Segment of the received Frame does not contain a Network Management vector nor a message ID 1_B Static segment: Network Management vector in the first part of the payload Dynamic segment: Message ID in the first part of the payload
RES	29	rh	**Reserved Bit** (vRF!Header!Reserved) Reflects the state of the received reserved bit. The reserved bit is transmitted as 0.
0	[15:11], [23:22], [31:30]	r	**Reserved** Returns 0 if read; should be written with 0.

Note: Header 3 is updated from Data Frames only.

Message Buffer Status (MBS)

The Message Buffer status is updated by the Communication Controller with respect to the assigned channel(s) latest at the end of the slot following the slot assigned to the Message Buffer. The flags are updated only when the Communication Controller is in “NORMAL_ACTIVE” or “NORMAL_PASSIVE” state. If only one channel (A or B) is assigned to a Message Buffer, the channel-specific status flags of the other channel are written to zero. If both channels are assigned to a Message Buffer, the channel-specific status flags of both channels are updated. The Message Buffer status is updated only when the slot counter reached the configured Frame ID and when the cycle counter filter matched. When the Host updates a Message Buffer via Input Buffer, all MBS flags are reset to zero independent of which IBCM bits are set or not. For details about receive / transmit filtering see **“Filtering and Masking” on Page 23-213**, **“Transmit Process” on Page 23-217**, and **“Receive Process” on Page 23-220**.

Whenever the Message Handler changes one of the flags VFRA, VFRB, SEOA, SEOB, CEOA, CEOB, SVOA, SVOB, TCIA, TCIB, ESA, ESB, MLST, FTA, FTB the respective Message Buffer’s MBC flag in registers MBSC1 to MBSC4 is set

MBS

Message Buffer Status **($070C_H$)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		RES	PPIS	NFIS	SYNS	SFIS	RCIS	0		CCS					
r		rh	rh	rh	rh	rh	rh	r		rh					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FTB	FTA	0	MLST	ESB	ESA	TCIB	TCIA	SVOB	SVOA	CEOB	CEOA	SEOB	SEOA	VRFB	VRFA
rh	rh	r	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
VFRA	0	rh	**Valid Frame Received on Channel A** (vSS!ValidFrameA) A valid Frame indication is set if a valid Frame was received on channel A. 0_B No valid Frame received on channel A 1_B Valid Frame received on channel A
VFRB	1	rh	**Valid Frame Received on Channel B** (vSS!ValidFrameB) A valid Frame indication is set if a valid Frame was received on channel B. 0_B No valid Frame received on channel B 1_B Valid Frame received on channel B

Field	Bits	Type	Description
SEOA	2	rh	**Syntax Error Observed on Channel A** (vSS!SyntaxErrorA) A syntax error was observed in the assigned slot on channel A. 0_B No syntax error observed on channel A 1_B Syntax error observed on channel A
SEOB	3	rh	**Syntax Error Observed on Channel B** (vSS!SyntaxErrorB) A syntax error was observed in the assigned slot on channel B. 0_B No syntax error observed on channel B 1_B Syntax error observed on channel B
CEOA	4	rh	**Content Error Observed on Channel A** (vSS!ContentErrorA) A content error was observed in the assigned slot on channel A. 0_B No content error observed on channel A 1_B Content error observed on channel A
CEOB	5	rh	**Content Error Observed on Channel B** (vSS!ContentErrorB) A content error was observed in the assigned slot on channel B. 0_B No content error observed on channel B 1_B Content error observed on channel B
SVOA	6	rh	**Slot Boundary Violation Observed on Channel A** (vSS!BViolationA) A slot boundary violation (channel active at the start or at the end of the assigned slot) was observed on channel A. 0_B No slot boundary violation observed on channel A 1_B Slot boundary violation observed on channel A
SVOB	7	rh	**Slot Boundary Violation Observed on Channel B** (vSS!BViolationB) A slot boundary violation (channel active at the start or at the end of the assigned slot) was observed on channel B. 0_B No slot boundary violation observed on channel B 1_B Slot boundary violation observed on channel B
TCIA	8	rh	**Transmission Conflict Indication Channel A** (vSS!TxConflictA) A transmission conflict indication is set if a transmission conflict has occurred on channel A. 0_B No transmission conflict occurred on channel A 1_B Transmission conflict occurred on channel A

Field	Bits	Type	Description
TCIB	9	rh	**Transmission Conflict Indication Channel B** (vSS!TxConflictB) A transmission conflict indication is set if a transmission conflict has occurred on channel B. 0_B No transmission conflict occurred on channel B 1_B Transmission conflict occurred on channel B
ESA	10	rh	**Empty Slot Channel A** In an empty slot there is no activity detected on the bus. The condition is checked in static and dynamic slots. 0_B Bus activity detected in the assigned slot on channel A 1_B No bus activity detected in the assigned slot on channel A
ESB	11	rh	**Empty Slot Channel B** In an empty slot there is no activity detected on the bus. The condition is checked in static and dynamic slots. 0_B Bus activity detected in the assigned slot on channel B 1_B No bus activity detected in the assigned slot on channel B
MLST	12	rh	**Message Lost** The flag is set in case the Host did not read the message before the Message Buffer was updated from a received Data Frame. Not affected by reception of NULL Frames except for Message Buffers belonging to the receive FIFO. The flag is reset by a Host write to the Message Buffer via IBF or when a new message is stored into the Message Buffer after the Message Buffers ND flag was reset by reading out the Message Buffer via OBF. 0_B No message lost 1_B Unprocessed message was overwritten
FTA	14	rh	**Frame Transmitted on Channel A** Indicates that this node has transmitted a Data Frame in the assigned slot on channel A. 0_B No transmission transmitted on channel A 1_B Data Frame transmitted on channel A in cycle defined by CCS bit field *Note: The FlexRay™ protocol specification requires that FTA can only be reset by the Host. Therefore the Cycle Count Status CCS for these bits is only valid for the cycle where the bits are set to 1*

Field	Bits	Type	Description
FTB	15	rh	**Frame Transmitted on Channel B** Indicates that this node has transmitted a Data Frame in the assigned slot on channel B. 0_B No transmission transmitted on channel B 1_B Data Frame transmitted on channel B in cycle defined by CCS bit field *Note: The FlexRay™ protocol specification requires that FTB can only be reset by the Host. Therefore the Cycle Count Status CCS for these bits is only valid for the cycle where the bits are set to 1*
CCS	[21:16]	rh	**Cycle Count Status** Cycle Count when status (MBS register) has been updated.
RCIS	24	rh	**Received on Channel Indicator Status** (vSS!Channel) Indicates the channel on which the Frame was received. 0_B Frame received on channel B 1_B Frame received on channel A *Note: For receive buffers (CFG = 0) the RCIS is updated from both valid data and NULL Frames. If no valid Frame was received, the previous value is maintained. For transmit buffers the flags have no meaning and should be ignored.*
SFIS	25	rh	**Startup Frame Indicator Status** (vRF!Header!SuFIndicator) A Startup Frame is marked by the Startup Frame indicator. 0_B No Startup Frame received 1_B The received Frame is a startup Frame *Note: For receive buffers (CFG = 0) the SFIS is updated from both valid data and NULL Frames. If no valid Frame was received, the previous value is maintained. For transmit buffers the flags have no meaning and should be ignored.*
SYNS	26	rh	**SYNC Frame Indicator Status** (vRF!Header!SyFIndicator) A Startup Frame is marked by the Startup Frame indicator. 0_B No SYNC Frame received 1_B The received Frame is a SYNC Frame *Note: For receive buffers (CFG = 0) the SYNS is updated from both valid data and NULL Frames. If no valid Frame was received, the previous value is maintained. For transmit buffers the flags have no meaning and should be ignored.*

Field	Bits	Type	Description
NFIS	27	rh	**NULL Frame Indicator Status** (vRF!Header!NFIndicator) If reset to 0 the Payload Segment of the received Frame contains no usable data. 0_B Received Frame is a NULL Frame 1_B Received Frame is not a NULL Frame *Note: For receive buffers (CFG = 0) the NFIS is updated from both valid data and NULL Frames. If no valid Frame was received, the previous value is maintained. For transmit buffers the flags have no meaning and should be ignored.*
PPIS	28	rh	**Payload Preamble Indictor Status** (vRF!Header!PPIndicator) The payload preamble indicator defines whether a Network Management vector or message ID is contained within the Payload Segment of the received Frame. 0_B Static Segment: The Payload Segment of the received Frame does not contain a Network Management vector or a message ID Dynamic Segment: The Payload Segment of the received Frame does not contain a Network Management vector or a message ID 1_B Static Segment: Network Management vector at the beginning of the payload Dynamic Segment: Message ID at the beginning of the payload *Note: For receive buffers (CFG = 0) the PPIS is updated from both valid data and NULL Frames. If no valid Frame was received, the previous value is maintained. For transmit buffers the flags have no meaning and should be ignored.*
RESS	29	rh	**Reserved Bit Status** (vRF!Header!Reserved) Reflects the state of the received reserved bit. The reserved bit is transmitted as 0. *Note: For receive buffers (CFG = 0) the RESS is updated from both valid data and NULL Frames. If no valid Frame was received, the previous value is maintained. For transmit buffers the flags have no meaning and should be ignored.*

Field	Bits	Type	Description
0	13, [23:22], [31:30]	r	**Reserved** Returns 0 if read; should be written with 0.

Output Buffer Command Mask (OBCM)

Configures how the Output Buffer is updated from the Message Buffer in the Message RAM selected by the Output Buffer Command Request register. If OBF Host and OBF Shadow are swapped, also mask bits OBCM.RDSH and OBCM.RHSH are swapped with bits OBCM.RDSS and OBCM.RHSS to keep them attached to the respective Output Buffer transfer.

OBCM
Output Buffer Command Mask **(0710_H)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0														RDSH	RHSH
r														rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														RDSS	RHSS
r														rwh	rwh

Field	Bits	Type	Description
RHSS	0	rwh	**Read Header Section Shadow** 0_B Header Section is not read 1_B Header Section selected for transfer from Message RAM to Output Buffer
RDSS	1	rwh	**Read Data Section Shadow** 0_B Data Section is not read 1_B Data Section selected for transfer from Message RAM to Output Buffer
RHSH	16	rh	**Read Header Section Host** 0_B Header Section is not read 1_B Header Section selected for transfer from Message RAM to Output Buffer
RDSH	17	rh	**Read Data Section Host** 0_B Data Section is not read 1_B Data Section selected for transfer from Message RAM to Output Buffer
0	[15:2], [31:18]	r	**Reserved** Returns 0 if read; should be written with 0.

Note: After the transfer of the Header Section from the Message RAM to OBF Shadow has completed, the Message Buffer status changed flag MBCn (n = 0-31) to MBCn (n = 96-127) of the selected Message Buffer in the Message Buffer Changed MBSC1 to MBSC4 registers is cleared. After the transfer of the Data Section from the Message RAM to OBF Shadow has completed, the New Data flag NDn (n = 0-31) to NDn (n = 96-127) of the selected Message Buffer in the New Data NDAT1 to NDAT4 registers is cleared.

Output Buffer Command Request (OBCR)

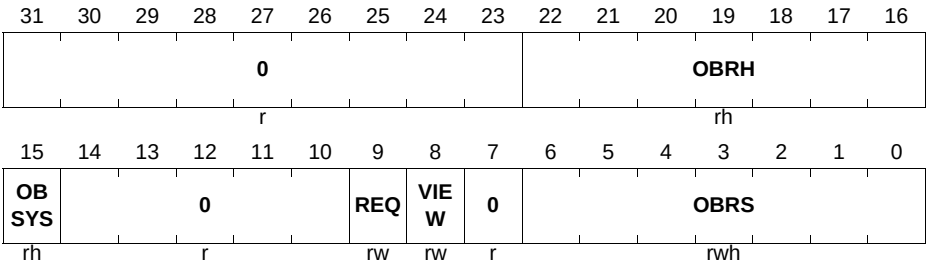
The Message Buffer selected by OBCR.OBRS is transferred from the Message RAM to the Output Buffer as soon as the Host has set OBCR.REQ. Bit OBCR.REQ can only be set while OBCR.OBSYS is 0 (see also **“Data Transfer from Message RAM to Output Buffer” on Page 23-226**).

After setting OBCR.REQ, OBCR.OBSYS is automatically set, and the transfer of the Message Buffer selected by OBCR.OBRS from the Message RAM to Output Buffer Shadow is started. When the transfer between the Message RAM and OBF Shadow has completed, this is signalled by clearing OBCR.OBSYS. By setting OBCR.VIEW while OBCR.OBSYS is 0, OBF Host and OBF Shadow are swapped. When Output Buffer Host and Output Buffer Shadow are swapped, also mask bits OBCM.RDSH and OBCM.RHSH are swapped with bits OBCM.RDSS and OBCM.RHSS to keep them attached to the respective Output Buffer transfer. Now the Host can read the transferred Message Buffer from OBF Host. In parallel the Message Handler may transfer the next message from the Message RAM to OBF Shadow if OBCR.VIEW and OBCR.REQ are set at the same time.

Any write access to an Output Buffer register while OBCR.OBSYS is set will cause the error flag EIR.IOBA to be set. In this case the Output Buffer will not be changed.

OBCR

Output Buffer Command Request (0714_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
OBRS	[6:0]	rw	**Output Buffer Request Shadow** Number of source Message Buffer to be transferred from the Message RAM to OBF Shadow. Valid values are 00_H to $7F_H$ (0 to 127). If the number of the first Message Buffer of the receive FIFO is written to this register the Message Handler transfers the Message Buffer addressed by the GET Index Register (GIDX, **"FIFO Function" on Page 23-221**) to OBF Shadow.
VIEW	8	rw	**View Shadow Buffer** Toggles between OBF Shadow and OBF Host. Only writeable while OBCR.OBSYS = 0. 0_B No action 1_B Swap OBF Shadow and OBF Host
REQ	9	rw	**Request Message RAM Transfer** Requests transfer of Message Buffer addressed by OBCR.OBRS from Message RAM to OBF Shadow. Only writeable while OBCR.OBSYS = 0. 0_B No request 1_B Transfer to OBF Shadow requested
OBSYS	15	rh	**Output Buffer Busy Shadow** Set to 1 after setting bit OBCR.REQ. When the transfer between the Message RAM and OBF Shadow has completed, OBCR.OBSYS is cleared again. 0_B No transfer in progress 1_B Transfer between Message RAM and OBF Shadow in progress

Field	Bits	Type	Description
OBRH	[22:16]	rh	**Output Buffer Request Host** Number of Message Buffer currently accessible by the Host via RDHS1 to RDHS3, MBS, and RDDSnn (nn = 01-64). By setting OBCR.VIEW OBF Shadow and OBF Host are swapped and the transferred Message Buffer is accessible by the Host. Valid values are 00_H to $7F_H$ (01 to 27).
0	7, [14:10], [31:23]	r	**Reserved** Returns 0 if read; should be written with 0.

23.6 Functional Description

This chapter describes the E-Ray implementation together with the related FlexRay™ protocol features. More information about the FlexRay™ protocol itself can be found in the FlexRay™ protocol specification v2.1.

Communication on FlexRay™ networks is based on Frames and symbols. The wakeup symbol (WUS) and the collision avoidance symbol (CAS) are transmitted outside the communication cycle to setup the time schedule. Frames and media access test symbols (MTS) are transmitted inside the communication cycle.

23.6.1 Communication Cycle

A communication cycle in FlexRay™ consists of the following elements:

- Static Segment
- Dynamic Segment
- Symbol Window
- Network Idle Time (NIT)

Static segment, dynamic segment, and symbol window form the Network Communication Time (NCT). For each communication channel the slot counter starts at 1 and counts up until the end of the dynamic segment is reached. Both channels share the same arbitration grid which means that they use the same synchronized Macrotick.

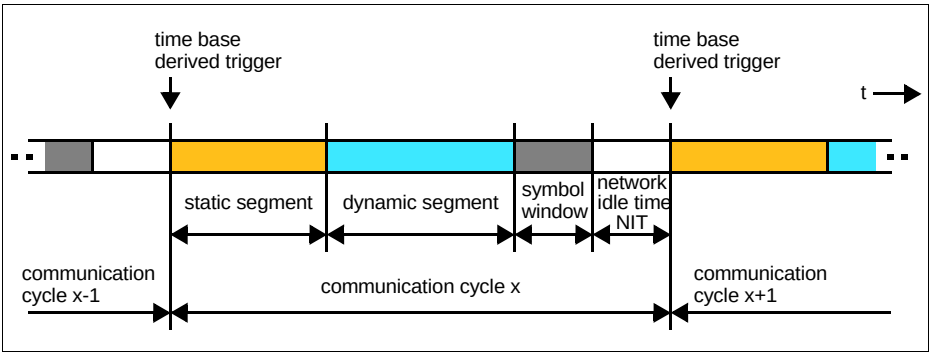


Figure 23-4 Structure of Communication Cycle

23.6.1.1 Static Segment

The Static Segment is characterized by the following features:

- Time slots of fixed length (optionally protected by bus guardian)
- Start of Frame transmission at action point of the respective static slot
- Payload length same for all Frames on both channel

Parameters: Number of Static Slots **GTUC07.NSS**, Static Slot Length **GTUC07.SSL**, Payload Length Static **MHDC.SFDL**, Action Point Offset **GTUC09.APO**.

23.6.1.2 Dynamic Segment

The Dynamic Segment is characterized by the following features:

- All controllers have bus access (no bus guardian protection possible)
- Variable payload length and duration of slots, different for both channels
- Start of transmission at minislot action point

Parameters: Number of Minislots **GTUC08.NMS**, Minislot Length **GTUC08.MSL** Minislot Action Point Offset **GTUC09.MAPO**, Start of Latest Transmit (last minislot) **MHDC.SLT**.

23.6.1.3 Symbol Window

During the symbol window only one media access test symbol (MTS) may be transmitted per channel. MTS symbols are send in “NORMAL_ACTIVE” state to test the bus guardian.

The symbol window is characterized by the following features:

- Send single symbol
- Transmission of the MTS symbol starts at the symbol windows action point

Parameters: Symbol Window Action Point Offset **GTUC09.APO** (same as for static slots), Network Idle Time Start **GTUC04.NIT**.

23.6.1.4 Network Idle Time (NIT)

During network idle time the Communication Controller has to perform the following tasks:

- Calculate clock correction terms (offset and rate)
- Distribute offset correction over multiple Macroticks
- Perform cluster cycle related tasks

Parameters: Network Idle Time Start **GTUC04.NIT**, Offset Correction Start **GTUC04.OCS**.

23.6.1.5 Configuration of Network Idle Time (NIT) Start and Offset Correction Start

The number of Macroticks per cycle (gMacroPerCycle) is assumed to be m. It is configured by programming **GTUC02.MPC** = m.

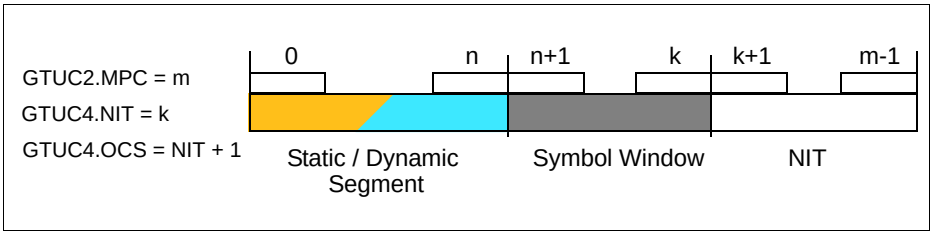


Figure 23-5 Configuration of network idle time (NIT) start and offset correction start

The static / dynamic segment starts with Macrotick 0 and ends with Macrotick n:
n = static segment length + dynamic segment offset + dynamic segment length - 1 Macrotick

n = gNumberOfStaticSlots • gdStaticSlot + dynamic segment offset
+ gNumberOfMinislots • gdMinislot - 1 Macroticks

The static segment length is configured by **GTUC07.SSL** and **GTUC07.NSS**.
The dynamic segment length is configured by **GTUC08.MSL** and **GTUC08.NMS**.

The dynamic segment offset is:

If gdActionPointOffset ≤ gdMinislotActionPointOffset:

dynamic segment offset = 0 MT

Else if gdActionPointOffset > gdMinislotActionPointOffset:

dynamic segment offset = gdActionPointOffset - gdMinislotActionPointOffset

The network idle time (NIT) starts with Macrotick k+1 and ends with the last Macrotick of cycle m-1. It has to be configured by setting **GTUC04.NIT** = k.

For the E-Ray the offset correction start is required to be

GTUC04.OCS ≥ **GTUC04.NIT** + 1 = k+1.

The length of symbol window results from the number of Macroticks between the end of the static / dynamic segment and the beginning of the NIT. It can be calculated by k - n.

23.6.2 Communication Modes

The FlexRay™ Protocol Specification v2.1 defines the Time-Triggered Distributed (TT-D) mode.

Time-triggered Distributed (TT-D)

In TT-D mode the following configurations are possible:

- **Pure static**: minimum 2 static slots + symbol window (optional)
- **Mixed static/dynamic**: minimum 2 static slots + dynamic segment + symbol window (optional)

A minimum of two coldstart nodes need to be configured for distributed time-triggered operation. Two fault-free coldstart nodes are necessary for the cluster startup. Each Startup Frame must be a SYNC Frame, therefore all coldstart nodes are sync nodes.

23.6.3 Clock Synchronization

In TT-D mode a distributed clock synchronization is used. Each node individually synchronizes itself to the cluster by observing the timing of received SYNC Frames from other nodes.

23.6.3.1 Global Time

Activities in a FlexRay™ node, including communication, are based on the concept of a global time, even though each individual node maintains its own view of it. It is the clock synchronization mechanism that differentiates the FlexRay™ cluster from other node collections with independent clock mechanisms. The global time is a vector of two values; the cycle (cycle counter) and the cycle time (Macrotick counter).

Cluster specific:

- Macrotick = basic unit of time measurement in a FlexRay™ network, a Macrotick consists of an integer number of Microticks
- Cycle length = duration of a communication cycle in units of Macroticks

23.6.3.2 Local Time

Internally, nodes time their behavior with Microtick resolution. Microticks are time units derived from the oscillator clock tick of the specific node. Therefore Microticks are controller-specific units. They may have different duration in different controllers. The precision of a node's local time difference measurements is a Microtick.

Node specific:

- Oscillator clock → prescaler → Microtick
- Microtick = basic unit of time measurement in a Communication Controller, clock correction is done in units of Microticks
- Cycle counter + Macrotick counter = nodes local view of the global time

23.6.3.3 Synchronization Process

Clock synchronization is performed by means of SYNC Frames. Only preconfigured nodes (sync nodes) are allowed to send SYNC Frames. In a two-channel cluster a sync node has to send its SYNC Frame on both channels.

For synchronization in FlexRay™ the following constraints have to be considered:

- Max. one SYNC Frame per node in one communication cycle
- Max. 15 SYNC Frames per cluster in one communication cycle
- Every node has to use all available SYNC Frames for clock synchronization
- Minimum of two sync nodes required for clock synchronization and startup

For clock synchronization the time difference between expected and observed arrival time of SYNC Frames received during the static segment, valid on both channels (two-channel cluster), is measured. The calculation of correction terms is done during network idle time (NIT) (offset: every cycle, rate: odd cycle) by using a FTA / FTM algorithm. For details see FlexRay™ protocol specification v2.1, chapter 8.

Offset (phase) Correction

- Only deviation values measured and stored in the current cycle used
- For a two channel node the smaller value will be taken
- Calculation during network idle time (NIT) of **every** communication cycle, value may be negative
- Offset correction value calculated in even cycles used for error checking only
- Checked against limit values (violation: “NORMAL_ACTIVE” → “NORMAL_PASSIVE” → “HALT”)
- Correction value is an integer number of Microticks
- Correction done in **odd** numbered cycles, distributed over the Macroticks beginning at offset correction start up to cycle end (end of network idle time (NIT)) to shift nodes next start of cycle (Macroticks lengthened / shortened)

Rate (frequency) Correction

- Pairs of deviation values measured and stored in even / odd cycle pair used
- For a two channel node the average of the differences from the two channels is used
- Calculated during network idle time (NIT) of **odd** numbered cycles, value may be negative
- Cluster drift damping is performed using global damping value

- Checked against limit values
- Correction value is a signed integer number of Microticks
- Distributed over Macroticks comprising the next **even / odd** cycle pair (Macroticks lengthened / shortened)

Synchronization Process

Clock synchronization is performed by means of SYNC Frames. Only preconfigured nodes (sync nodes) are allowed to send SYNC Frames. In a two-channel cluster a sync node has to send its SYNC Frame on both channels.

For synchronization in FlexRay™ the following constraints have to be considered:

- Max. one SYNC Frame per node in one communication cycle
- Max. 15 SYNC Frames per cluster in one communication cycle
- Every node has to use all available SYNC Frames for clock synchronization
- Minimum of two sync nodes required for clock synchronization and startup

For clock synchronization the time difference between expected and observed arrival time of SYNC Frames received during the static segment, valid on both channels (two-channel cluster), is measured. The calculation of correction terms is done during network idle time (NIT) (offset: every cycle, rate: odd cycle) by using a FTA / FTM algorithm. For details see FlexRay™ protocol specification v2.1, chapter 8.

SYNC Frame Transmission

SYNC Frame transmission is only possible from buffer 0 and 1. Message Buffer 1 may be used for SYNC Frame transmission in case that SYNC Frames should have different payloads on the two channels. In this case bit **MRC.SPLM** has to be programmed to 1.

Message Buffers used for SYNC Frame transmission have to be configured with the key slot ID and can be (re)configured in “DEFAULT_CONFIG” or “CONFIG” state only. For nodes transmitting SYNC Frames **SUCC1.TXSY** must be set to 1.

23.6.3.4 External Clock Synchronization

During normal operation, independent clusters can drift significantly. If synchronous operation across independent clusters is desired, external synchronization is necessary; even though the nodes within each cluster are synchronized. This can be accomplished with synchronous application of host-deduced rate and offset correction terms to the clusters.

- External offset / rate correction value is a signed integer
- External offset / rate correction value is added to calculated offset / rate correction value
- Aggregated offset / rate correction term (external + internal) is not checked against configured limits

23.6.4 Error Handling

The implemented error handling concept is intended to ensure that in case of a lower layer protocol error in a single node communication between non-affected nodes can be maintained. In some cases, higher layer program command activity is required for the Communication Controller to resume normal operation. A change of the error handling state will set bit **EIR.PEMC** in the Error Service Request Register and may trigger an service request to the Host if enabled. The actual error mode is signalled by **CCEV.ERRM** in the Communication Controller Error Vector register.

Table 23-10 Error Modes of the POC (Degradation Model)

Error Mode	Activity
ACTIVE (green)	**Full operation**, State: “NORMAL_ACTIVE” The Communication Controller is fully synchronized and supports the cluster wide clock synchronization. The host is informed of any error condition(s) or status change by interrupt (if enabled) or by reading the error and status interrupt flags from registers **EIR** and **SIR**.
PASSIVE (yellow)	**Reduced operation**, State: “NORMAL_PASSIVE”, Communication Controller self rescue allowed The Communication Controller stops transmitting Frames and symbols, but received Frames are still processed. Clock synchronization mechanisms are continued based on received Frames. No active contribution to the cluster wide clock synchronization. The host is informed of any error condition(s) or status change by interrupt (if enabled) or by reading the error and status interrupt flags from registers **EIR** and **SIR**.
COMM_HALT (red)	**Operation halted**, State: “HALT”, Communication Controller self rescue not allowed The Communication Controller stops Frame and symbol processing, clock synchronization processing, and the Macrotick generation. The host has still access to error and status information by reading the error and status interrupt flags from registers **EIR** and **SIR**. The bus drivers are disabled.

23.6.4.1 Clock Correction Failed Counter

When the Clock Correction Failed Counter reaches the maximum “without clock correction passive” limit defined by **SUCC3.WCP**, the POC transits from “NORMAL_ACTIVE” to “NORMAL_PASSIVE” state. When it reaches the “maximum without clock correction fatal” limit defined by **SUCC3.WCF**, it transits “NORMAL_ACTIVE” or “NORMAL_PASSIVE” to the “HALT” state. Both limits are defined in the SUC Configuration Register 3.

The Clock Correction Failed Counter **CCEV.CCFC** allows the Host to monitor the duration of the inability of a node to compute clock correction terms after the Communication Controller passed protocol startup phase. It will be incremented by one at the end of any **odd** numbered communication cycle where either the Missing Offset Correction signal **SFS.MOCS** nor the Missing Rate Correction signal **SFS.MRCS** flag is set. The two flags are located in the SYNC Frame Status register, while the Clock Correction Failed Counter is located in the Communication Controller Error Vector register.

The Clock Correction Failed Counter is reset to zero at the end of an **odd** communication cycle if neither the Missing Offset Correction signal **SFS.MOCS** nor the Missing Rate Correction signal **SFS.MRCS** flag is set.

The Clock Correction Failed Counter stops incrementing when the “maximum without clock correction fatal” value **SUCC3.WCF** as defined in the SUC Configuration Register 3 is reached (i.e. incrementing the counter at its maximum value will not cause it to “wraparound” back to zero). The Clock Correction Failed Counter is initialized to zero when the Communication Controller enters “READY” state or when “NORMAL_ACTIVE” state is entered.

23.6.4.2 Passive to Active Counter

The passive to active counter controls the transition of the POC from “NORMAL_PASSIVE” to “NORMAL_ACTIVE” state. **SUCC1.PTA** in the SUC Configuration Register 1 defines the number of consecutive even / odd cycle pairs that must have valid clock correction terms before the Communication Controller is allowed to transit from “NORMAL_PASSIVE” to “NORMAL_ACTIVE” state. If **SUCC1.PTA** is reset to zero the Communication Controller is not allowed to transit from “NORMAL_PASSIVE” to “NORMAL_ACTIVE” state.

23.6.4.3 HALT Command

In case the Host wants to stop FlexRay™ communication of the local node it can bring the Communication Controller into “HALT” state by asserting the HALT command. This can be done by writing **SUCC1.CMD** = 0110_B in the SUC Configuration Register 1. When called in “NORMAL_ACTIVE” or “NORMAL_PASSIVE” state the POC transits to “HALT” state at the end of the current cycle. When called in any other state **SUCC1.CMD** will be reset to 0000_B = “COMMAND_NOT_ACCEPTED” and bit **EIR.CNA** in the Error Service Request Register is set to 1. If enabled an service request to the Host is generated.

23.6.4.4 FREEZE Command

In case the Host detects a severe error condition it can bring the Communication Controller into “HALT” state by asserting the FREEZE command. This can be done by writing **SUCC1.CMD** = 0111_B in the SUC Configuration Register 1. The FREEZE

command triggers the entry of the “HALT” state immediately regardless of the actual POC state.

The POC state from which the transition to HALT state took place can be read from **CCSV.PSL**.

23.6.5 Communication Controller States

This chapter introduces the states of the Communication Controller.

23.6.5.1 Communication Controller State Diagram

State transitions are controlled by externals the application reset or RXDA/B, by the POC state machine, and by the CHI Command Vector **SUCC1.CMD** located in the SUC Configuration Register 1.

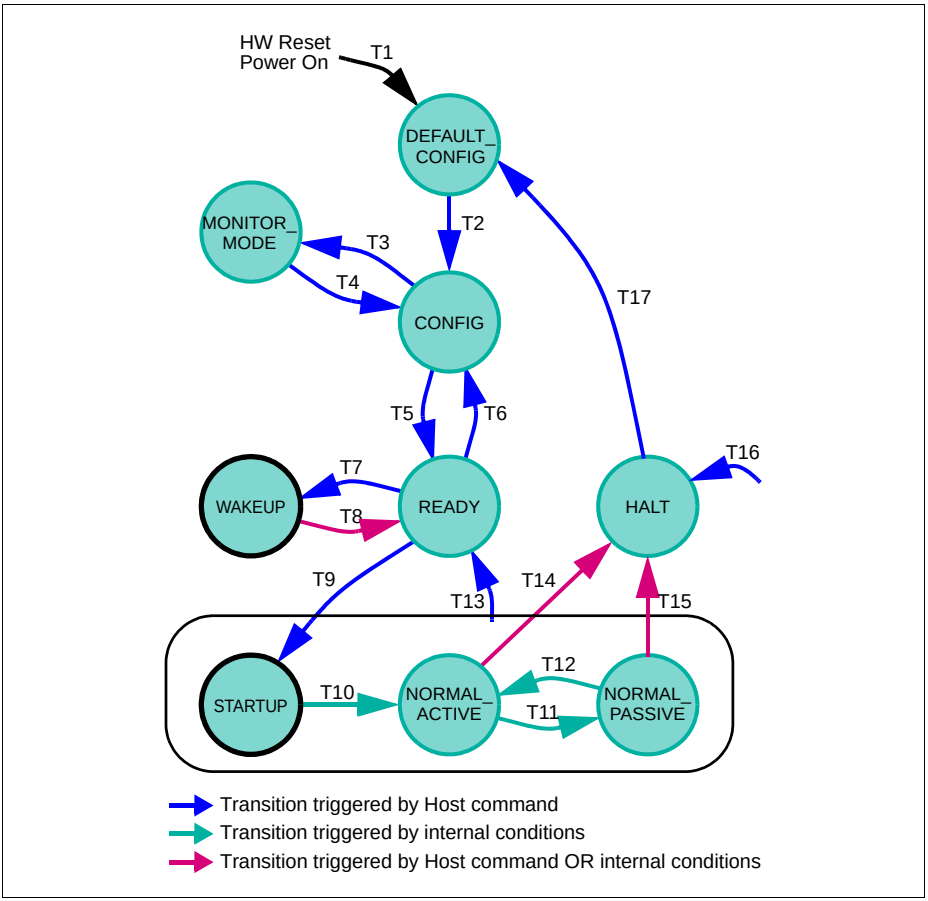


Figure 23-6 Overall State Diagram of E-Ray Communication Controller

The Communication Controller exits from all states to “HALT” state after application of the FREEZE command (SUCC1.CMD = 0111_B).

Table 23-11 State Transitions of E-Ray Overall State Machine

T#	Condition	From	To
1	application reset	HW Reset	DEFAULT_CONFIG
2	Command CONFIG, SUCC1.CMD = 0001_B	DEFAULT_CONFIG	CONFIG
3	Unlock sequence followed by command MONITOR_MODE, SUCC1.CMD = 1011_B	CONFIG	MONITOR_MODE
4	Command CONFIG, SUCC1.CMD = 0001_B	MONITOR_MODE	CONFIG
5	Unlock sequence followed by command READY, SUCC1.CMD = 0010_B	CONFIG	READY
6	Command CONFIG, SUCC1.CMD = 0001_B	READY	CONFIG
7	Command WAKEUP, SUCC1.CMD = 0011_B	READY	WAKEUP
8	Complete, non-aborted transmission of wakeup pattern OR received WUP OR received Frame Header OR command READY, SUCC1.CMD = 0010_B	WAKEUP	READY
9	Command RUN, SUCC1.CMD = 0100_B	READY	STARTUP
10	Successful startup	STARTUP	NORMAL_ACTIVE
11	Clock Correction Failed counter reached Maximum Without Clock Correction Passive limit configured by WCP in SUC Configuration Register 3	NORMAL_ACTIVE	NORMAL_PASSIVE
12	Number of valid correction terms reached the Passive to Active limit configured by PTA in SUC Configuration Register 1	NORMAL_PASSIVE	NORMAL_ACTIVE
13	Command READY, SUCC1.CMD = 0010_B	STARTUP, NORMAL_ACTIVE, NORMAL_PASSIVE	READY

Table 23-11 State Transitions of E-Ray Overall State Machine (cont'd)

T#	Condition	From	To
14	Clock Correction Failed counter reached Maximum Without Clock Correction Fatal limit configured by WCF in SUC Configuration Register 3 AND bit HCSE in the SUC Configuration Register 1 set to 1 OR command HALT, SUCC1.CMD = 0110_B	NORMAL_ACTIVE	HALT
15	Clock Correction Failed counter reached Maximum Without Clock Correction Fatal limit configured by WCF in SUC Configuration Register 3 AND bit HCSE in the SUC Configuration Register 1 set to 1 OR command HALT, SUCC1.CMD = 0110_B	NORMAL_PASSIVE	HALT
16	Command FREEZE, SUCC1.CMD = 0111_B	All States	HALT
17	Command CONFIG, SUCC1.CMD = 0001_B	HALT	DEFAULT_CONFIG

23.6.5.2 DEFAULT_CONFIG State

In "DEFAULT_CONFIG" state, the Communication Controller is stopped. All configuration registers are accessible and the pins to the physical layer are in their inactive state.

The Communication Controller enters this state

- When leaving application reset
- When exiting from "HALT" state

To leave "DEFAULT_CONFIG" state the Host has to write SUCC1.CMD = 0001_B in the SUC Configuration Register 1. The Communication Controller transits to "CONFIG" state.

CONFIG State

In "CONFIG" state, the Communication Controller is stopped. All configuration registers are accessible and the pins to the physical layer are in their inactive state. This state is used to initialize the Communication Controller configuration.

The Communication Controller enters this state

- When exiting from “DEFAULT_CONFIG” state
- When exiting from “MONITOR_MODE” or “READY” state

When the state has been entered via “HALT” and “DEFAULT_CONFIG” state, the Host can analyze status information and configuration. Before leaving “CONFIG” state the Host has to assure that the configuration is fault-free.

To leave “CONFIG” state, the Host has to perform the unlock sequence as described on **“LCK” on Page 23-30**. Directly after unlocking the “CONFIG” state the Host has to write **SUCC1.CMD** in the SUC Configuration Register 1 to enter the next state.

Internal counters and the Communication Controller status flags are reset when the Communication Controller leaves “CONFIG”.

*Note: The Message Buffer Status Registers (**MHDS**, **TXRQ1** to **TXRQ4**, **NDAT1** to **NDAT4**, **MBSC1** to **MBSC4**) and status data stored in the Message RAM and are not affected by the transition of the POC from “CONFIG” to “READY” state.*

When the Communication Controller is in “CONFIG” state it is also possible to bring the Communication Controller into a power saving mode by halting the module clocks (f_{SCLK}, f_{CLC_ERAY}). To do this the Host has to assure that all Message RAM transfers have finished before turning off the clocks.

23.6.5.3 MONITOR_MODE

After unlocking “CONFIG” state and writing **SUCC1.CMD** = 0011_B the Communication Controller enters “MONITOR_MODE”. In this mode the Communication Controller is able to receive FlexRay™ Frames and to detect wakeup pattern. The temporal integrity of received Frames is not checked, and therefore cycle counter filtering is not supported. It is not possible to distinguish between static and dynamic frames, because limited functions in Monitor Mode (FRF.RSS will be ignored, filtering not functional).This mode can be used for debugging purposes in case e.g. that startup of a FlexRay™ network fails. After writing **SUCC1.CMD** = 0001_B the Communication Controller transits back to “CONFIG” state.

In MONITOR_MODE the pick first valid mechanism is disabled. This means that a receive Message Buffer may only be configured to receive on one channel. Received Frames are stored into Message Buffers according to Frame ID and receive channel. NULL Frames are handled like Data Frames. After Frame reception only status bits **MBS.VFRA**, **MBS**, **MBS.MLST**, **MBS.RCIS**, **MBS.SFIS**, **MBS.SYNS**, **MBS.NFIS**, **MBS.PPIS**, **MBS.RESS** have valid value.

In “MONITOR_MODE” the Communication Controller is not able to distinguish between CAS and MTS symbols. In case one of these symbols is received on one or both of the two channels, the flags **SIR.MTSA** resp. **SIR.MTSB** are set. **SIR.CAS** has no function in “MONITOR_MODE”.

23.6.5.4 READY State

After unlocking “CONFIG” state and writing **SUCC1.CMD** = 0010_B the Communication Controller enters “READY” state. From this state the Communication Controller can transit to WAKEUP state and perform a cluster wakeup or to “STARTUP” state to perform a coldstart or to integrate into a running communication.

The Communication Controller enters this state

- When exiting from “CONFIG”, “WAKEUP”, “STARTUP”, “NORMAL_ACTIVE”, or “NORMAL_PASSIVE” state by writing **SUCC1.CMD** = 0010_B (READY command).

The Communication Controller exits from this state

- To “CONFIG” state by writing **SUCC1.CMD** = 0001_B (CONFIG command)
- To “WAKEUP” state by writing **SUCC1.CMD** = 0011_B (WAKEUP command)
- To “STARTUP” state by writing **SUCC1.CMD** = 0100_B (RUN command)

Internal counters and the Communication Controller status flags are reset when the Communication Controller enters “STARTUP” state.

*Note: Status bits **MHDS**, registers **TXRQ1** to **TXRQ4**, and status data stored in the Message RAM are not affected by the transition of the POC from “READY” to “STARTUP” state.*

23.6.5.5 WAKEUP State

The description below is intended to help configuring wakeup for the E-Ray IP-module. A detailed description of the wakeup procedure together with the respective SDL diagrams can be found in the FlexRay™ protocol specification v2.1, section 7.1.

The Communication Controller enters this state

- When exiting from “READY” state by writing **SUCC1.CMD** = 0011_B (WAKEUP command).

The Communication Controller exits from this state to “READY” state

- After complete non-aborted transmission of wakeup pattern
- After WUP reception
- After detecting a WUP collision
- After reception of a Frame Header
- By writing **SUCC1.CMD** = 0010_B (READY command)

The cluster wakeup must precede the communication startup in order to ensure that all mechanisms defined for the startup work properly. The minimum requirement for a cluster wakeup is that all bus drivers are supplied with power. A bus driver has the ability to wake up the other components of its node when it receives a wakeup pattern on its channel. At least one node in the cluster needs an **external** wakeup source.

The Host completely controls the wakeup procedure. It is informed about the state of the cluster by the bus driver and the Communication Controller and configures bus guardian

(if available) and Communication Controller to perform the cluster wakeup. The Communication Controller provides to the Host the ability to transmit a special wakeup pattern on each of its available channels separately. The Communication Controller needs to recognize the wakeup pattern only during “WAKEUP” state.

Wakeup may be performed on only one channel at a time. The Host has to configure the wakeup channel while the Communication Controller is in “CONFIG” state by writing bit **SUCC1.WUCS** in the SUC Configuration Register 1. The Communication Controller ensures that ongoing communication on this channel is not disturbed. The Communication Controller cannot guarantee that all nodes connected to the configured channel awake upon the transmission of the wakeup pattern, since these nodes cannot give feedback until the startup phase. The wakeup procedure enables single-channel devices in a two-channel system to trigger the wakeup, by only transmitting the wakeup pattern on the single channel to which they are connected. Any coldstart node that deems a system startup necessary will then wake the remaining channel before initiating communication startup.

The wakeup procedure tolerates any number of nodes simultaneously trying to wakeup a single channel and resolves this situation such that only one node transmits the pattern. Additionally the wakeup pattern is collision resilient, so even in the presence of a fault causing two nodes to simultaneously transmit a wakeup pattern, the resulting collided signal can still wake the other nodes.

After wakeup the Communication Controller returns to “READY” state and signals the change of the wakeup status to the Host by setting bit **SIR.WST** in the Status Service Request Register. The wakeup status vector can be read from the Communication Controller Status Vector register **CCSV.WSV**. If a valid wakeup pattern was received also either flag **SIR.WUPA** or flag **SIR.WUPB** in the Status Service Request Register is set.

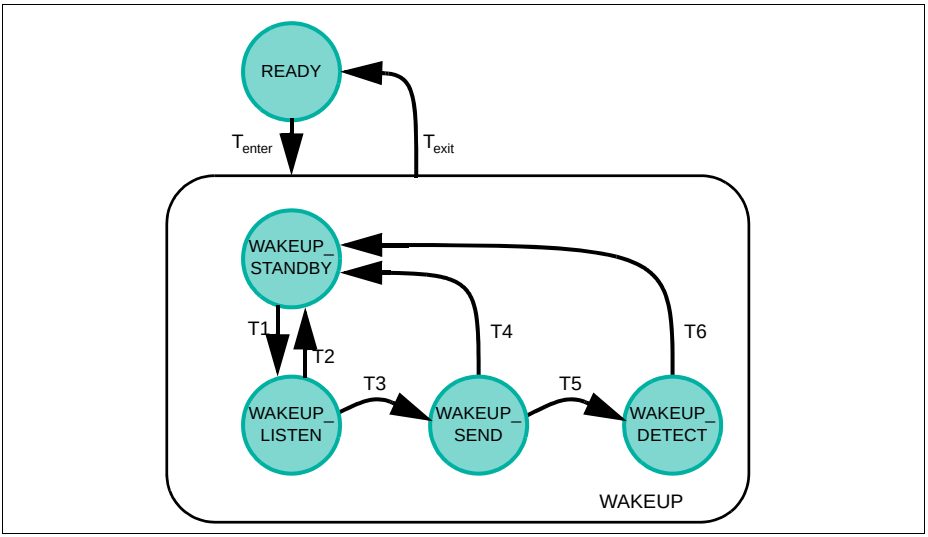


Figure 23-7 Structure of POC State WAKEUP

Table 23-12 State Transitions WAKEUP

T#	Condition	From	To
enter	Host commands change to “WAKEUP” state by writing **SUCC1.CMD** = 0011_B (WAKEUP command)	READY	WAKEUP
1	CHI command WAKEUP triggers wakeup FSM to transit to “WAKEUP_LISTEN” state	WAKEUP_ STANDBY	WAKEUP_ LISTEN
2	Received WUP on wakeup channel selected by flag **SUCC1.WUCS** in the SUC Configuration Register 1 OR Frame Header on either available channel	WAKEUP_ LISTEN	WAKEUP_ STANDBY
3	Timer event	WAKEUP_ LISTEN	WAKEUP_ SEND
4	Complete, non-aborted transmission of wakeup pattern	WAKEUP_ SEND	WAKEUP_ STANDBY

Table 23-12 State Transitions WAKEUP (cont'd)

T#	Condition	From	To
5	Collision detected	WAKEUP_ SEND	WAKEUP_ DETECT
6	Wakeup timer expired OR WUP detected on wakeup channel selected by flag **SUCC1.WUCS** in the SUC Configuration Register 1 OR Frame Header received on either available channel	WAKEUP_ DETECT	WAKEUP_ STANDBY
exit	Wakeup completed (after T2 or T4 or T6) OR Host commands change to "READY" state by writing **SUCC1.CMD** = 0010_B (READY command). This command also resets the wakeup FSM to "WAKEUP_STANDBY" state	WAKEUP	READY

The "WAKEUP_LISTEN" state is controlled by the wakeup timer and the wakeup noise timer. The two timers are controlled by the parameters listen time-out **SUCC2.LT** and listen time-out noise **SUCC2.LTN**. Both values can be configured in the SUC Configuration Register 2. listen time-out enables a fast cluster wakeup in case of a noise free environment, while listen time-out noise enables wakeup under more difficult conditions regarding noise interference.

In "WAKEUP_SEND" state the Communication Controller transmits the wakeup pattern on the configured channel and checks for collisions. After return from wakeup the Host has to bring the Communication Controller into "STARTUP" state by CHI command RUN.

In "WAKEUP_DETECT" state the Communication Controller attempts to identify the reason for the wakeup collision detected in "WAKEUP_SEND" state. The monitoring is bounded by the expiration of listen time-out as configured by **SUCC2.LT** in the SUC Configuration Register 2. Either the detection of a wakeup pattern indicating a wakeup attempt by another node or the reception of a Frame Header indication existing communication, causes the direct transition to "READY" state. Otherwise WAKEUP_DETECT is left after expiration of listen time-out; in this case the reason for wakeup collision is unknown.

The Host has to be aware of possible failures of the wakeup and act accordingly. It is advisable to delay any potential startup attempt of the node having instigated the wakeup by the minimal time it takes another coldstart node to become awake and to be configured.

The FlexRay™ Protocol Specification v2.1 recommends that two different Communication Controllers shall awake the two channels.

Host activities

The host must coordinate the wakeup of the two channels and must decide whether, or not, to wake a specific channel. The sending of the wakeup pattern is initiated by the Host and generated by the Communication Controller. The wakeup pattern is detected by the remote BDs and signalled to their local Hosts.

Wakeup procedure controlled by Host (single-channel wakeup):

- Configure the Communication Controller in “CONFIG” state
  - Select wakeup channel by programming bit **SUCC1.WUCS**
- Check local BDs whether a WUP was received
- Activate BD of selected wakeup channel
- Command Communication Controller to start wakeup on the configured channel by writing **SUCC1.CMD** = 0011_B
  - Communication Controller enters “WAKEUP
  - Communication Controller returns to “READY” state and signals status of wakeup attempt to Host
- Wait predefined time to allow the other nodes to wakeup and configure themselves
- Coldstart node: wait for WUP on the other channel
  - In a dual channel cluster wait for WUP on the other channel
  - Reset coldstart inhibit flag **CCSV.CSI** by writing **SUCC1.CMD** = 1001_B (ALLOW_COLDSTART command))
- Reset Coldstart Inhibit flag **CCSV.CSI** in the CCSV register by writing **SUCC1.CMD** = 1001_B (ALLOW_COLDSTART command), coldstart node only
- Command Communication Controller to enter startup by writing **SUCC1.CMD** = 0100_B (RUN command)

Wakeup procedure triggered by BD:

- Wakeup recognized by BD
- BD triggers power-up of Host (if required)
- BD signals wakeup event to Host
- Host configures its local Communication Controller
- If necessary Host commands wakeup of second channel and waits predefined time to allow the other nodes to wakeup and configure themselves
- Host commands Communication Controller to enter “STARTUP” state by writing **SUCC1.CMD** = 0100_B (RUN command)

Wakeup pattern (WUP)

The wakeup pattern is composed of at least two wakeup symbols (WUS). Wakeup symbol and wakeup pattern are configured by the PRT Configuration Registers **PRTC1** and **PRTC2**.

- Single channel wakeup, wakeup symbol may not be sent on both channels at the same time

- Wakeup symbol collision resilient for up to two sending nodes (two overlapping wakeup symbols still recognizable)
- Wakeup symbol must be configured identical in all nodes of a cluster
- Wakeup symbol transmit low time configured by **PRTC2.TXL**
- Wakeup symbol idle time used to listen for activity on the bus, configured by **PRTC2.TXI**
- A wakeup pattern composed of at least two Tx-wakeup symbols needed for wakeup
- Number of repetitions configurable by **PRTC1.RWP** (2 to 63 repetitions)
- Wakeup symbol receive window length configured by **PRTC1.RXW**
- Wakeup symbol receive low time configured by **PRTC2.RXL**
- Wakeup symbol receive idle time configured by **PRTC2.RXI**

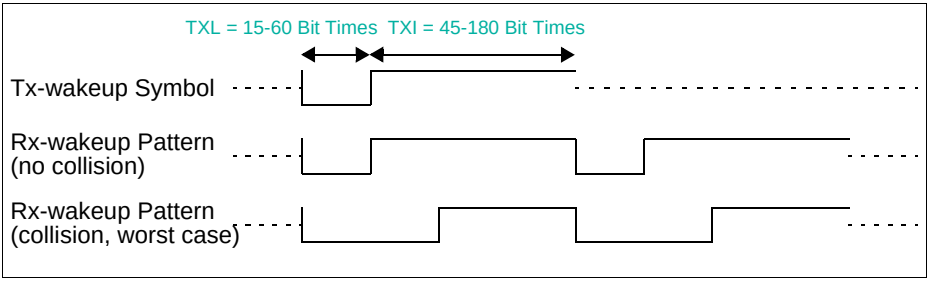


Figure 23-8 Timing of Wakeup Pattern

23.6.5.6 STARTUP State

The description below is intended to help configuring startup for the E-Ray IP-module. A detailed description of the startup procedure together with the respective SDL diagrams can be found in the FlexRay™ protocol specification v2.1, section 7.2.

Any node entering “STARTUP” state that has coldstart capability should assure that both channels attached have been awakened before initiating coldstart.

It cannot be assumed that all nodes and stars need the same amount of time to become completely awake and to be configured. Since at least two nodes are necessary to start up the cluster communication, it is advisable to delay any potential startup attempt of the node having instigated the wakeup by the minimal amount of time it takes another coldstart node to become awake, to be configured and to enter startup. It may require several hundred milliseconds (depending on the hardware used) before all nodes and stars are completely awakened and configured.

Startup is performed on all channels synchronously. During startup, a node only transmits startup Frames.

A fault-tolerant, distributed startup strategy is specified for initial synchronization of all nodes. In general, a node may enter “NORMAL_ACTIVE” state via (see **Figure 23-9**):

- Coldstart path initiating the schedule synchronization (leading coldstart node)
- Coldstart path joining other coldstart nodes (following coldstart node)
- Integration path integrating into an existing communication schedule (all other nodes)

A coldstart attempt begins with the transmission of a collision avoidance symbol (CAS). Only a coldstart node that had transmitted the CAS transmits Frames in the first four cycles after the CAS, it is then joined firstly by the other coldstart nodes and afterwards by all other nodes.

A coldstart node has the Transmit SYNC Frame in Key Slot bits **SUCC1.TXST** and **SUCC1.TXSY** in the SUC Configuration Register 1 set to 1. The Message Buffer 0 holds the key slot ID which defines the slot number where the Startup Frame is send. In the Frame Header of the Startup Frame the Startup Frame indicator bit is set.

In clusters consisting of three or more nodes, at least three nodes shall be configured to be coldstart nodes. In clusters consisting of two nodes, both nodes must be coldstart nodes. At least two fault-free coldstart nodes are necessary for the cluster to startup.

Each Startup Frame must also be a SYNC Frame; therefore each coldstart node will also be a sync node. The number of coldstart attempts is configured by **SUCC1.CSA** in the SUC Configuration Register 1.

A non-coldstart node requires at least two startup Frames from distinct nodes for integration. It may start integration before the coldstart nodes have finished their startup. It will not finish its startup until at least two coldstart nodes have finished their startup.

Both non-coldstart nodes and coldstart nodes start passive integration via the integration path as soon as they receive SYNC Frames from which to derive the TDMA schedule information. During integration the node has to adapt its own clock to the global clock (rate and offset) and has to make its cycle time consistent with the global schedule observable at the network. Afterwards, these settings are checked for consistency with all available network nodes. The node can only leave the integration phase and actively participate in communication when these checks are passed.

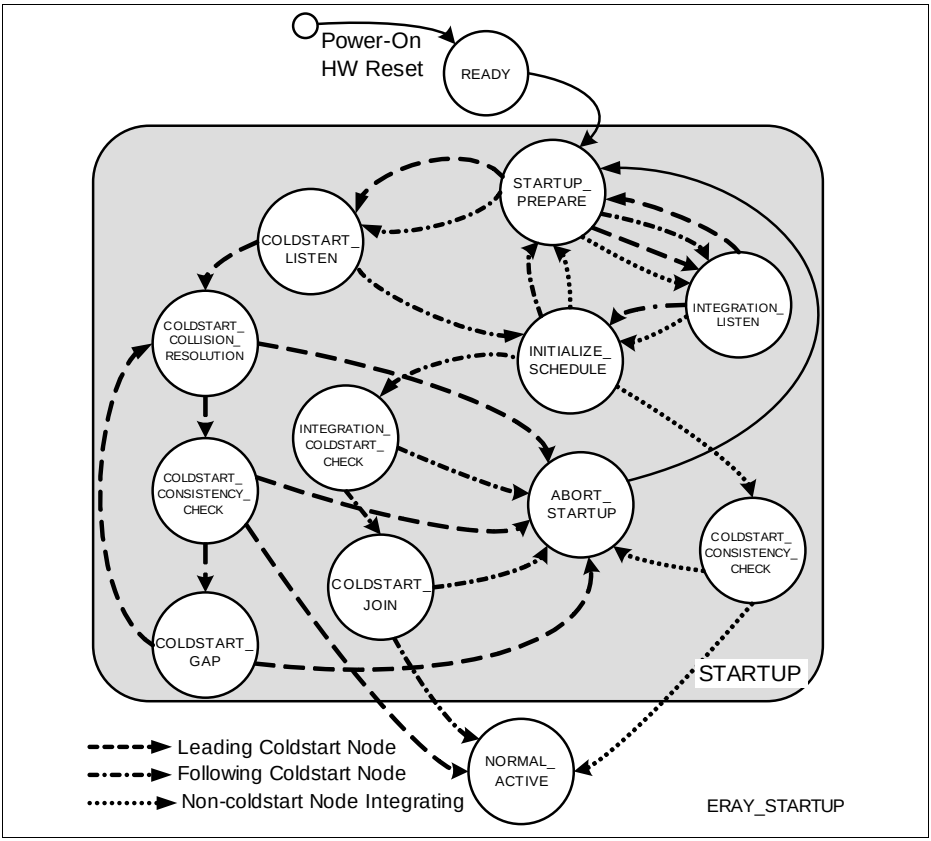


Figure 23-9 State Diagram Time-Triggered Startup

Coldstart Inhibit Mode

In coldstart inhibit mode the node is prevented from initializing the TDMA communication schedule. If bit **CCSV.CSI** in the Communication Controller Status Vector register is set, the node is not allowed to initialize the cluster communication, i.e. entering the coldstart path is prohibited. The node is allowed to integrate to a running cluster or to transmit startup Frames after another coldstart node started the initialization of the cluster communication.

The coldstart inhibit bit **CCSV.CSI** is set whenever the POC enters “READY” state. The bit has to be cleared under control of the Host by CHI command ALLOW_COLDSTART (**SUCC1.CMD** = 1001_B)

23.6.5.7 Startup Time-outs

The Communication Controller supplies two different Microtick timers supporting two time-out values, startup time-out and startup noise time-out. The two timers are reset when the Communication Controller enters the “COLDSTART_LISTEN” state. The expiration of either of these timers causes the node to leave the initial sensing phase (“COLDSTART_LISTEN” state) with the intention of starting up communication.

*Note: The startup and startup noise timers are identical with the wakeup and wakeup noise timers and use the same configuration values **SUCC2.LT** and **SUCC2.LTN** from the SUC Configuration Register 2.*

Startup Time-out

The startup time-out limits the listen time used by a node to determine if there is already communication between other nodes or at least one coldstart node actively requesting the integration of others.

The startup timer is configured by programming **SUCC2.LT** (pdListenTimeout) in the SUC Configuration Register 2.

The startup timer is restarted upon:

- Entering the “COLDSTART_LISTEN” state
- Both channels reaching idle state while in “COLDSTART_LISTEN” state

The startup timer is stopped:

- If communication channel activity is detected on one of the configured channels while the node is in the “COLDSTART_LISTEN” state
- When the “COLDSTART_LISTEN” state is left

Once the startup time-out expires, neither an overflow nor a cyclic restart of the timer is performed. The timer status is kept for further processing by the startup state machine.

Startup Noise Time-out

At the same time the startup timer is started for the first time (transition from “STARTUP_PREPARE” state to “COLDSTART_LISTEN” state), the startup noise timer is started. This additional time-out is used to improve reliability of the startup procedure in the presence of noise.

The startup noise timer is configured by programming **SUCC2.LTN** (gListenNoise - 1) in the SUC Configuration Register 2 (see **“SUC Configuration Register 2 (SUCC2)” on Page 23-87**).

The startup noise time-out is:
pdListenTimeout • ´gListenNoise = **SUCC2.LT** • (**SUCC2.LTN** + 1)

The startup noise timer is restarted upon:

- Entering the “COLDSTART_LISTEN” state
- Reception of correctly decoded Headers or CAS symbols while the node is in “COLDSTART_LISTEN” state

The startup noise timer is stopped when the “COLDSTART_LISTEN” state is left.

Once the startup noise time-out expires, neither an overflow nor a cyclic restart of the timer is performed. The status is kept for further processing by the startup state machine. Since the startup noise timer won’t be restarted when random channel activity is sensed, this time-out defines the fall-back solution that guarantees that a node will try to start up the communication cluster even in the presence of noise.

23.6.5.8 Path of leading Coldstart Node (initiating coldstart)

When a coldstart node enters “COLDSTART_LISTEN”, it listens to its attached channels.

If no communication is detected, the node enters the “COLDSTART_COLLISION_RESOLUTION” state and commences a coldstart attempt. The initial transmission of a CAS symbol is succeeded by the first regular cycle. This cycle has the number zero.

From cycle zero on, the node transmits its startup Frame. Since each coldstart node is allowed to perform a coldstart attempt, it may occur that several nodes simultaneously transmit the CAS symbol and enter the coldstart path. This situation is resolved during the first four cycles after CAS transmission.

As soon as a node that initiates a coldstart attempt receives a CAS symbol or a Frame Header during these four cycles, it re-enters the “COLDSTART_LISTEN” state. Thereby, only one node remains in this path. In cycle four, other coldstart nodes begin to transmit their startup Frames.

After four cycles in “COLDSTART_COLLISION_RESOLUTION” state, the node that initiated the coldstart enters the “COLDSTART_CONSISTENCY_CHECK” state. It collects all startup Frames from cycle four and five and performs the clock correction. If the clock correction does not deliver any errors and it has received at least one valid Startup Frame pair, the node leaves “COLDSTART_CONSISTENCY_CHECK” and enters “NORMAL_ACTIVE” state.

The number of coldstart attempts that a node is allowed to perform is configured by **SUCC1.CSA** in the SUC Configuration Register 1. The number of remaining coldstarts attempts **CCSV.RCA** can be read from Communication Controller Status Vector register. The number of remaining attempts is reduced by one for each attempted coldstart. A node may enter the “COLDSTART_LISTEN” state only if this value is larger than one and it may enter the “COLDSTART_COLLISION_RESOLUTION” state only if this value is larger than zero. If the number of coldstart attempts is one, coldstart is inhibited but integration is still possible.

Path of following Coldstart Node (responding to leading Coldstart Node)

When a coldstart node enters the “COLDSTART_LISTEN” state, it tries to receive a valid pair of startup Frames to derive its schedule and clock correction from the leading coldstart node.

As soon as a valid Startup Frame has been received the “INITIALIZE_SCHEDULE” state is entered. If the clock synchronization can successfully receive a matching second valid Startup Frame and can derive a schedule from this startup Frames, the “INTEGRATION_COLDSTART_CHECK” state is entered.

In “INTEGRATION_COLDSTART_CHECK” state it is assured that the clock correction can be performed correctly and that the coldstart node from which this node has initialized its schedule is still available. The node collects all SYNC Frames and performs clock correction in the following double-cycle. If clock correction does not signal any errors and if the node continues to receive sufficient Frames from the same node it has integrated on, the “COLDSTART_JOIN” state is entered.

In “COLDSTART_JOIN” state integrating coldstart nodes begin to transmit their own startup Frames. Thereby the node that initiated the coldstart and the nodes joining it can check if their schedules agree to each other. If for the following three cycles the clock correction does not signal errors and at least one other coldstart node is visible, the node leaves “COLDSTART_JOIN” state and enters “NORMAL_ACTIVE” state. Thereby it leaves “STARTUP” at least one cycle after the node that initiated the coldstart.

Path of Non-coldstart Node

When a non-coldstart node enters the INTEGRATION_LISTEN state, it listens to its attached channels and tries to receive FlexRay™ Frames.

As soon as a valid Startup Frame has been received the “INITIALIZE_SCHEDULE” state is entered. If the clock synchronization can successfully receive a matching second valid Startup Frame and derive a schedule from this, the INTEGRATION_CONSISTENCY_CHECK state is entered.

In “INTEGRATION_CONSISTENCY_CHECK” state it is verified that the clock correction can be performed correctly and that enough coldstart nodes (at least 2) send startup Frames that agree to the nodes own schedule. Clock correction is activated, and if any errors are signalled, the integration attempt is aborted.

During the first even cycle in this state, either two valid startup Frames or the Startup Frame of the node that this node has integrated on must be received; otherwise the node aborts the integration attempt.

During the first double-cycle in this state, either two valid Startup Frame pairs or the Startup Frame pair of the node that this node has integrated on must be received; otherwise the node aborts the integration attempt.

If after the first double-cycle less than two valid startup Frames are received within an even cycle, or less than two valid Startup Frame pairs are received within a double-cycle, the startup attempt is aborted.

Nodes in this state need to see two valid Startup Frame pairs for two consecutive double-cycles each to be allowed to leave STARTUP and enter NORMAL_OPERATION. Consequently, they leave startup at least one double-cycle after the node that initiated the coldstart and only at the end of a cycle with an odd cycle number.

23.6.5.9 NORMAL_ACTIVE State

As soon as the node that transmitted the first CAS symbol (resolving the potential access conflict and entering “STARTUP” via coldstart path) and one additional node have entered the “NORMAL_ACTIVE” state, the startup phase for the cluster has finished. In the “NORMAL_ACTIVE” state, all configured messages are scheduled for transmission. This includes all Data Frames as well as the SYNC Frames. Rate and offset measurement is started in all even cycles (even/odd cycle pairs required).

In “NORMAL_ACTIVE” state the Communication Controller supports regular communication functions

- The Communication Controller performs transmissions and reception on the FlexRay™ bus as configured
- Clock synchronization is running
- The Host interface is operational

The Communication Controller exits from that state to

- “HALT” state by writing **SUCC1.CMD** = 0110_B (HALT command, at the end of the current cycle)
- “HALT” state by writing **SUCC1.CMD** = 0111_B (FREEZE command, immediately)
- “HALT” state due to change of the error state from “ACTIVE” to “COMM_HALT”
- “NORMAL_PASSIVE” state due to change of the error state from “ACTIVE” to “PASSIVE”
- “READY” state by writing **SUCC1.CMD** = 0010_B (READY command)

23.6.5.10 NORMAL_PASSIVE State

“NORMAL_PASSIVE” state is entered from “NORMAL_ACTIVE” state when the error state changes from ACTIVE (green) to PASSIVE (yellow).

In “NORMAL_PASSIVE” state, the node is able to receive all Frames (node is fully synchronized and performs clock synchronization). In comparison to the “NORMAL_ACTIVE” state the node does not actively participate in communication, i.e. neither symbols nor Frames are transmitted.

In “NORMAL_PASSIVE” state

- The Communication Controller performs reception on the FlexRay™ bus
- The Communication Controller does not transmit any Frames or symbols on the FlexRay™ bus
- Clock synchronization is running
- The Host interface is operational

The Communication Controller exits from this state to

- “HALT” state by writing **SUCC1.CMD** = 0110_B (HALT command, at the end of the current cycle)
- “HALT” state by writing **SUCC1.CMD** = 0111_B (FREEZE command, immediately)
- “HALT” state due to change of the error state from “PASSIVE” to “COMM_HALT”
- “NORMAL_ACTIVE” state due to change of the error state from “PASSIVE” to “ACTIVE”. The transition takes place when **CCEV.PTAC** from the Communication Controller Error Vector register equals **SUCC1.PTA** - 1.
- “READY” state by writing **SUCC1.CMD** = 0010_B (READY command)

23.6.5.11 HALT State

In this state all communication (reception and transmission) is stopped.

The Communication Controller enters this state

- By writing **SUCC1.CMD** = 0110_B (HALT command) while the Communication Controller is in “NORMAL_ACTIVE” or “NORMAL_PASSIVE” state
- By writing **SUCC1.CMD** = 0111_B (FREEZE command) from all states
- When exiting from “NORMAL_ACTIVE” state because the clock correction failed counter reached the “maximum without clock correction fatal” limit
- When exiting from “NORMAL_PASSIVE” state because the clock correction failed counter reached the “maximum without clock correction fatal” limit

The Communication Controller exits from this state to “CONFIG” state

- By writing **SUCC1.CMD** = 0001_B (DEFAULT_CONFIG command)

When the Communication Controller enters “HALT” state, all configuration and status data is maintained for analyzing purposes.

When the Host writes **SUCC1.CMD** = 0110_B (HALT command) in the SUC Configuration Register 1 to 1, the Communication Controller sets bit **CCSV.HRQ** in the Communication Controller Status Vector register and enters “HALT” state after the current communication cycle has finished.

When the Host writes **SUCC1.CMD** = 0111_B (FREEZE command) in the SUC Configuration Register to 1, the Communication Controller enters “HALT” state immediately and sets the **CCSV.FSI** bit in the Communication Controller Status Vector register.

The POC state from which the transition to HALT state took place can be read from **CCSV.PSL**.

23.6.6 Network Management

The accrued Network Management (NM) vector is located in the Network Management Register 1 to Network Management Register 3 (**NMVx (x = 1-3)**). The Communication Controller performs a logical OR operation over all Network Management (NM) vectors out of all received valid Network Management (NM) Frames with the Payload Preamble Indicator (PPI) bit set. Only a static Frame may be configured to hold Network Management (NM) information. The Communication Controller updates the Network Management (NM) vector at the end of each cycle.

The length of the Network Management (NM) vector can be configured from 0 to 12 byte by NML in the NEM Configuration Register. The Network Management (NM) vector length must be configured identically in all nodes of a cluster.

To configure a transmit buffer to send FlexRay™ Frames with the PPI bit set, the PPIT bit in the Header Section of the respective transmit buffer has to be set via **WRHS1.PPIT**. In addition the Host has to write the Network Management (NM) information to the Data Section of the respective transmit buffer.

The evaluation of the Network Management (NM) vector has to be done by the application running on the Host.

*Note: In case a Message Buffer is configured for transmission / reception of Network Management Frames, the payload length configured in Header 2 of that Message Buffer should be equal or greater than the length of the NM Vector configured by **NEMC.NML**.*

When the Communication Controller transits to “HALT” state, the cycle count is not incremented and therefore the NM Vector is not updated. In this case NMV1 to NMV3 holds the value from the cycle before.

23.6.7 Filtering and Masking

Filtering is done by checking specific fields in a received Frame against the corresponding configuration constants of the valid Message Buffers and the actual slot and cycle counter values (acceptance filtering), or by comparing the configuration constants of the valid Message Buffers against the actual slot and cycle counter values (transmit filtering). A Message Buffer is only updated / transmitted if the required matches occur.

Filtering is done on the following fields:

- Channel ID
- Frame ID
- Cycle Counter

The following filter combinations for acceptance / transmit filtering are allowed:

- Frame ID + Channel ID
- Frame ID + Channel ID + Cycle Counter

In order to store a received message in a Message Buffer all configured filters must match.

Note: For the FIFO the acceptance filter is configured by the FIFO Rejection Filter and the FIFO Rejection Filter Mask.

A message will be transmitted in the time slot corresponding to the configured Frame ID on the configured channel(s). If cycle counter filtering is enabled the configured cycle filter value must also match.

23.6.7.1 Frame ID Filtering

Every transmit and receive buffer contains a Frame ID stored in the Header Section. This Frame ID is used differently for receive and transmit buffers.

Receive Buffers

A received message is stored in the first receive buffer where the received Frame ID matches the configured Frame ID, provided channel ID and cycle counter criteria are also met.

Transmit Buffers

For transmit buffers the configured Frame ID is used to determine the appropriate slot for message transmission. The Frame will be transmitted in the time slot corresponding to the configured Frame ID, provided channel ID and cycle counter criteria are also met.

23.6.7.2 Channel ID Filtering

There is a 2-bit channel filtering field (CHA, CHB) located in the Header Section of each Message Buffer in the Message RAM. It serves as a filter for receive buffers, and as a control field for transmit buffers (see **Table 23-13**).

Table 23-13 Channel Filtering Configuration

CHA	CHB	Transmit Buffer transmit Frame	Receive Buffer store valid receive Frame
1	1	on both channels (static segment only)	received on channel A or B (store first semantically valid Frame, static segment only)
1	0	on channel A	received on channel A
0	1	on channel B	received on channel B
0	0	no transmission	ignore Frame

Note: If a Message Buffer is configured for the dynamic segment and both bits of the channel filtering field are set to 1, no Frames are transmitted resp. received Frames are ignored (same function as CHA = CHB = 0)

Receive Buffers

Valid received Frames are stored if they are received on the channels specified in the channel filtering field. Only in static segment a receive buffer may be setup for reception on both channels (CHA and CHB set). Other filtering criteria must also be met.

If a valid Header Segment was stored, the respective MBC flag in the Message Buffer Status Changed register is set. If a valid Payload Segment was stored, the respective **NDn (n = 0-31)** to **NDn (n = 96-127)** flag in the New Data **NDAT1** to **NDAT4** register is set. In both cases, if bit **RDHS1.MBI** in the Header Section of the respective Message Buffer is set, the RXI flag in the Status Service Request Register is set to 1. If enabled an service request is generated.

Transmit Buffers

The content of the buffer is transmitted only on the channels specified in the channel filtering field when the Frame ID filtering and cycle counter filtering criteria are also met. Only in static segment a transmit buffer may be setup for transmission on both channels (CHA and CHB set). After transmission has completed, and if bit **WRHS1.MBI** in the Header Section of the respective Message Buffer is set, the TXI flag in the Status Service Request Register is set to 1. If enabled an service request is generated.

23.6.7.3 Cycle Counter Filtering

Cycle counter filtering is based on the notion of a cycle set. For filtering purposes, a match is detected if any one of the elements of the cycle set is matched. The cycle set is defined by the cycle code field in the Header Section of each Message Buffer.

If Message Buffer 0 is configured to hold the startup / SYNC Frame or the single slot Frame by bits TXST, TXSY, and TSM in the SUC Configuration Register 1, cycle counter filtering for Message Buffer 0 should be disabled.

*Note: Sharing of a static time slot via cycle counter filtering between different nodes of a FlexRay™ network is **not** allowed.*

The set of cycle numbers belonging to a cycle set is determined as described in **Table 23-14**.

Table 23-14 Definition of Cycle Set

Cycle Code	Matching Cycle Counter Values		
$000000x_B$	all Cycles		
$000001c_B$	every second Cycle	at (Cycle Count)mod2	= c
$00001cc_B$	every fourth Cycle	at (Cycle Count)mod4	= cc
$0001ccc_B$	every eighth Cycle	at (Cycle Count)mod8	= ccc
$001cccc_B$	every sixteenth Cycle	at (Cycle Count)mod16	= cccc
$01ccccc_B$	every thirty-second Cycle	at (Cycle Count)mod32	= ccccc
$1cccccc_B$	every sixty-fourth Cycle	at (Cycle Count)mod64	= cccccc

Table 23-15 below gives some examples for valid cycle sets to be used for cycle counter filtering:

Table 23-15 Examples for Valid Cycle Sets

Cycle Code	Matching Cycle Counter Values
0000011_B	1-3-5-7- ….-63 ↵
0000100_B	0-4-8-12- ….-60 ↵
0001110_B	6-14-22-30- ….-62 ↵
0011000_B	8-24-40-56 ↵
0100011_B	3-35 ↵
1001001_B	9 ↵

Receive Buffers

The received message is stored only if the received cycle counter matches an element of the receive buffer's cycle set. Channel ID and Frame ID criteria must also be met.

Transmit Buffers

The content of the buffer is transmitted on the configured channels when an element of the cycle set matches the current cycle counter value and the Frame ID matches the slot counter value.

23.6.7.4 FIFO Filtering

For FIFO filtering there is one rejection filter and one rejection filter mask available. The FIFO rejection filter consists of 20 bits for **Channel** (2 bits), **Frame ID** (11 bits), and **Cycle Code** (7 bits). Rejection filter and rejection filter mask can be configured in

DEFAIULT_CONGIF or “CONFIG” state only. The filter configuration in the Header Sections of Message Buffers belonging to the FIFO is ignored.

A valid received Frame is stored in the FIFO if channel ID, Frame ID, and cycle counter are not rejected by the configured rejection filter and rejection filter mask, and if there is no matching dedicated receive buffer.

23.6.8 Transmit Process

The transmit process is described in the following sections.

23.6.8.1 Static Segment

For the static segment, if there are several messages pending for transmission, the message with the Frame ID corresponding to the next sending slot is selected for transmission.

The Data Section of transmit buffers assigned to the static segment can be updated until the end of the preceding time slot. This means that a transfer from the Input Buffer has to be started by writing to the Input Buffer Command Request register latest at this time.

23.6.8.2 Dynamic Segment

In the dynamic segment, if several messages are pending, the message with the highest priority (lowest Frame ID) is selected next. Only Frame ID’s which are higher than the largest static Frame ID are allowed for the dynamic segment.

In the dynamic segment different slot counter sequences are possible (concurrent sending of different Frame ID’s on both channels). Therefore pending messages are selected according to their Frame ID and their channel configuration bit.

The Data Section of transmit buffers assigned to the dynamic segment can be updated until the end of the preceding slot. This means that a transfer from the Input Buffer has to be started by writing to the Input Buffer Command Request register latest at this time.

The start of latest transmit configured by SLT in the MHD Configuration Register 1 defines the maximum minislot value allowed before inhibiting new Frame transmission in the dynamic segment of the current cycle.

23.6.8.3 Transmit Buffers

A portion of the E-Ray Message Buffers can be configured as transmit buffers by programming bit CFG in the Header Section of the respective Message Buffer to 1. This can be done via the Write Header Section 1 register.

There exist the following possibilities to assign a transmit buffer to the Communication Controller channels:

- Static segment: channel A **or** channel B, channel A **and** channel B

- Dynamic segment: channel A **or** channel B

Message Buffer 0 is dedicated to hold the startup Frame, the SYNC Frame, or the designated single slot Frame as configured by TXST, TXSY, and TSM in the SUC Configuration Register 1. In this case it can be reconfigured in “DEFAULT_CONFIG” or “CONFIG” state only. This ensures that any node transmits at most one startup / SYNC Frame per communication cycle. Transmission of startup / SYNC Frames from other Message Buffers is not possible.

All other Message Buffers configured for transmission in static or dynamic segment are reconfigurable during runtime. Due to the organization of the Data Partition in the Message RAM (reference by data pointer), reconfiguration of the configured payload length and the data pointer in the Header Section of a Message Buffer may lead to erroneous configurations. If a Message Buffer is reconfigured during runtime it may happen that this Message Buffer is not send out in the respective communication cycle.

The Communication Controller does not have the capability to calculate the Header CRC. The Host is supposed to provide the Header CRCs for all transmit buffers. If Network Management is required the Host has to set the PPIT bit in the Header Section of the respective Message Buffer to 1 and write the Network Management information to the Data Section of the Message Buffer (see **Section 23.6.6**).

The payload length field configures the data payload length in 2-byte words. If the configured payload length of a static transmit buffer is shorter than the payload length configured for the static segment by SFDL in the Message Handler Configuration Register 1, the Communication Controller generates padding byte to ensure that Frames have proper physical length. The padding pattern is logical zero.

Each transmit buffer provides a transmission mode flag TXM that allows the Host to configure the transmission mode for the transmit buffer in the static segment. If this bit is set, the transmitter operates in the single-shot mode. If this bit is cleared, the transmitter operates in the continuous mode. In dynamic segment the transmitter always works in single-shot mode.

If a Message Buffer is configured in the continuous mode, the Communication Controller does not reset the transmission request flag TXR after successful transmission. In this case a Frame is sent out each time the Frame ID and cycle counter filter match. The TXR flag can be reset by the Host by writing the respective Message Buffer number to the Input Buffer Command Request register while bit **STXRH** in the Input Buffer Command Mask register is reset to 0.

If two or more transmit buffers are configured with the same Frame ID **and** cycle counter filter value, the transmit buffer with the lowest Message Buffer number will be transmitted in the respective slot.

23.6.8.4 Frame Transmission

To prepare a transmit buffer for transmission the following steps are required:

- Configure the Message Buffer as transmit buffer by writing bit CFG = 1 in the Write Header Section 1 register
- Write transmit message (Header and Data Section) to the Input Buffer.
- To transfer a transmit message from Input Buffer to the Message RAM proceed as described on **“Data Transfer from Input Buffer to Message RAM” on Page 23-224**.
- If configured in the Input Buffer Command Mask register the Transmission Request flag for the respective Message Buffer will be set as soon as the transfer has completed, and the Message Buffer is ready for transmission.
- Check whether the Message Buffer has been transmitted by checking the TXR bits (TXR = 0) in the Transmission Request 1,2 registers (single-shot mode only).

In single-shot mode the Communication Controller resets the TXR flag after transmission has been completed. Now the Host may update the transmit buffer with the next message. The Communication Controller does not transmit the message before the Host has indicated that the update is completed by setting the Transmission Request flag TXR again. The Host can check the actual state of the TXR flags of all Message Buffers by reading the Transmission Request registers. After successful transmission, if bit **WRHS1.MBI** in the Header Section of the respective Message Buffer is set, the transmit service request flag in the Status Service Request Register is set (TXI = 1). If enabled an service request is generated.

23.6.8.5 NULL Frame Transmission

If in static segment the Host does not set the transmission request flag before transmit time, and if there is no other transmit buffer with matching filter criteria (matching Frame ID and cycle counter filter), the Communication Controller transmits a NULL Frame with the NULL Frame indication bit reset to 0 and the payload data reset to zero.

In the following cases the Communication Controller transmits a NULL Frame with the NULL Frame indication bit reset to 0, and the rest of the Frame Header and the Frame length unchanged (payload data is reset to zero):

- All transmit buffers configured for the slot have cycle counter filters that do not match the current cycle
- There are matching Frame ID’s and cycle counter filters, but none of these transmit buffers has the transmission request flag TXR set

NULL Frames are not transmitted in the dynamic segment.

23.6.9 Receive Process

The receive process is described in the following sections.

23.6.9.1 Frame Reception

To prepare or change a Message Buffer for reception the following steps are required:

- Configure the Message Buffer as receive buffer by writing bit CFG = 0 in the Write Header Section 1 register
- Configure the receive buffer by writing the configuration data (Header Section) to the Input Buffer
- Transfer the configuration from Input Buffer to the Message RAM by writing the number of the target Message Buffer to the Input Buffer Command Request register.

Once these steps are performed, the Message Buffer functions as an active receive buffer and participates in the internal acceptance filtering process, which takes place every time the Communication Controller receives a message. The first matching receive buffer is updated from the received message. If the Message Buffer holds an unprocessed Data Section (ND = 1) it is overwritten with the new message and the MLST bit in the respective Message Buffer Status register is set.

If the payload length of a received Frame PLC is longer than the value programmed by PLC in the Header Section of the respective Message Buffer, the data field stored in the Message Buffer is truncated to that length.

If no Frame, a NULL Frame, or a corrupted Frame is received in a slot, the Data Section of the Message Buffer configured for this slot is not updated. In this case only the flags in the Message Buffer Status register are updated to signal the cause of the problem. In addition the respective MBC flag in the Message Buffer Status Changed 1,2,3,4 registers is set.

When the Data Section of a receive buffer has been updated from a received Frame, the respective New Data **NDn (n = 0-31)** to **NDn (n = 96-127)** flag in the New Data **NDAT1** to **NDAT4** registers is set. When the Message Handler has updated the Message Buffer status, the respective MBC flag in the Message Buffer Status Changed 1,2,3,4 registers is set. If bit **RDHS1.MBI** in the Header Section of the respective Message Buffer is set, the receive service request flag in the Status Service Request Register is set (RXI = 1). If enabled an service request is generated.

To read a receive buffer from the Message RAM via the Output Buffer proceed as described on **“Data Transfer from Message RAM to Output Buffer” on Page 23-226**.

Note: The ND and MBC flags are automatically cleared by the Message Handler when the received message has been transferred to the Output Buffer.

23.6.9.2 NULL Frame reception

The Payload Segment of a received NULL Frame is **not** copied into the matching receive buffer. If a NULL Frame has been received, the Header Section of the matching Message Buffer is updated from the received NULL Frame. The NULL Frame indication bit in the Header Section 3 of the respective Message Buffer is reset (NFI = 0) and the respective MBC flag in the Message Buffer Status Changed 1,2,3,4 registers is set.

In case that bit ND and / or MBC were already set before this event because the Host did not read the last received message, bit MLST in the Message Buffer Status register of the respective Message Buffer is also set.

23.6.10 FIFO Function

A group of the Message Buffers can be configured as a cyclic First-In-First-Out (FIFO). The group of Message Buffers belonging to the FIFO is contiguous in the register map starting with the Message Buffer referenced by FFB and ending with the Message Buffer referenced by LCB in the Message RAM Configuration register. Up to 128 Message Buffers can be assigned to the FIFO.

23.6.10.1 Description

Every valid incoming message not matching with any dedicated receive buffer but passing the programmable FIFO filter is stored into the FIFO. In this case Frame ID, payload length, receive cycle count, and the status bits of the addressed FIFO Message Buffer are overwritten with Frame ID, payload length, receive cycle count, and the status from the received message and can be read by the Host for message identification. Bit RFNE in the Status Service Request Register shows that the FIFO is not empty, bit RFF in the Status Service Request Register is set when the last available Message Buffer belonging to the FIFO is written, bit RFO in the Error Service Request Register shows that a FIFO overrun has been detected. If enabled, service requests are generated.

There are two index registers associated with the FIFO. The PUT Index Register (PIDX) is an index to the next available location in the FIFO. When a new message has been received it is written into the Message Buffer addressed by the PIDX register. The PIDX register is then incremented and addresses the next available Message Buffer. If the PIDX register is incremented past the highest numbered Message Buffer of the FIFO, the PIDX register is loaded with the number of the first (lowest numbered) Message Buffer in the FIFO chain. The GET Index Register (GIDX) is used to address the next Message Buffer of the FIFO to be read. The GIDX register is incremented after transfer of the contents of a Message Buffer belonging to the FIFO to the Output Buffer. The PUT Index Register and the GET Index Register are not accessible by the Host.

The FIFO is completely filled when the PUT index (PIDX) reaches the value of the GET index (GIDX). When the next message is written to the FIFO before the oldest message has been read, both PUT index and GET index are incremented and the new message

overwrites the oldest message in the FIFO. This will set FIFO overrun flag RFO in the Error Service Request Register.

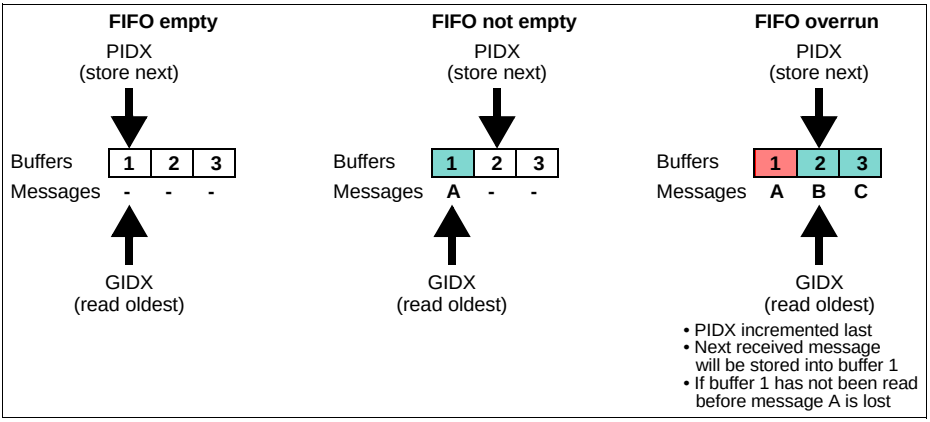


Figure 23-10 FIFO Status: Empty, Not Empty, Overrun

A FIFO non empty status is detected when the PUT index (PIDX) differs from the GET index (GIDX). In this case flag RFNE is set. This indicates that there is at least one received message in the FIFO. The FIFO empty, FIFO not empty, and the FIFO overrun states are explained in **Figure 23-10** for a three Message Buffer FIFO.

There is a programmable FIFO rejection filter for the FIFO. The FIFO Rejection Filter register (FRF) defines a filter pattern for messages to be rejected. The FIFO rejection filter consists of channel filter, Frame ID filter, and cycle counter filter. If bit RSS is set to 1 (default), all messages received in the static segment are rejected by the FIFO. If bit RNF is set to 1 (default), received NULL Frames are not stored in the FIFO.

The FIFO Rejection Filter Mask register (FRFM) specifies which bits of the Frame ID filter in the FIFO Rejection Filter register are marked “don’t care” for rejection filtering.

23.6.10.2 Configuration of the FIFO

For all Message Buffers belonging to the FIFO the data pointer to the first 32-bit word of the Data Section of the respective Message Buffer in the Message RAM has to be configured via the Write Header Section 3 register. All information required for acceptance filtering is taken from the FIFO rejection filter and the FIFO rejection filter mask and needs not be configured in the Header Sections of the Message Buffers belonging to the FIFO.

When programming the data pointers for the Message Buffers belonging to the FIFO, the payload length of all Message Buffers should be programmed to the same value.

*Note: It is recommended to program the MBI bits of the Message Buffers belonging to the FIFO to 0 via **WRHS1.MBI** to avoid generation of RX interrupts.*
*If the payload length of a received Frame is longer than the value programmed by **WRHS2.PLC** in the Header Section of the respective Message Buffer, the data field stored in a Message Buffer of the FIFO is truncated to that length.*

23.6.10.3 Access to the FIFO

To read from the FIFO the Host has to trigger a transfer from the Message RAM to the Output Buffer by writing the number of the first Message Buffer of the FIFO (referenced by FFB) to the Output Buffer Command Request register. The Message Handler then transfers the Message Buffer addressed by the GET Index Register (GIDX) to the Output Buffer. After this transfer the GET Index Register (GIDX) is incremented.

23.6.11 Message Handling

The Message Handler controls data transfers between the Input / Output Buffer and the Message RAM and between the Message RAM and the two Transient Buffer RAMs. All accesses to the internal RAM's are 32 bit accesses.

Access to the Message Buffers stored in the Message RAM is done under control of the Message Handler state machine. This avoids conflicts between accesses of the two protocol controllers and the Host to the Message RAM.

Frame IDs of Message Buffers assigned to the static segment have to be in the range from 1 to NSS as configured in the GTU Configuration Register 7. Frame IDs of Message Buffers assigned to the dynamic segment have to be in the range from NSS + 1 to 2047.

Received messages with no matching dedicated receive buffer (static or dynamic segment) are stored in the receive FIFO (if configured) if they pass the FIFO rejection filter.

23.6.11.1 Host access to Message RAM

The message transfer between Input Buffer and Message RAM as well as between Message RAM and Output Buffer is triggered by the Host by writing the number of the target / source Message Buffer to be accessed to the Input or Output Buffer Command Request register.

The Input / Output Buffer Command Mask registers can be used to write / read Header and Data Section of the selected Message Buffer separately. If bit **STXRS** in the Input Buffer Command Mask register is set (**STXRS** = 1), the transmission request flag TXR of the selected Message Buffer is automatically set after the Message Buffer has been updated.

If bit **STXRS** in the Input Buffer Command Mask register is reset (**STXRS** = 0), the transmission request flag TXR of the selected Message Buffer is reset. This can be used to stop transmission from Message Buffers operated in continuous mode.

Input Buffer (IBF) and the Output Buffer (OBF) are build up as a double buffer structure. One half of this double buffer structure is accessible by the Host (IBF Host / OBF Host), while the other half (IBF Shadow / OBF Shadow) is accessed by the Message Handler for data transfers between IBF / OBF and Message RAM.

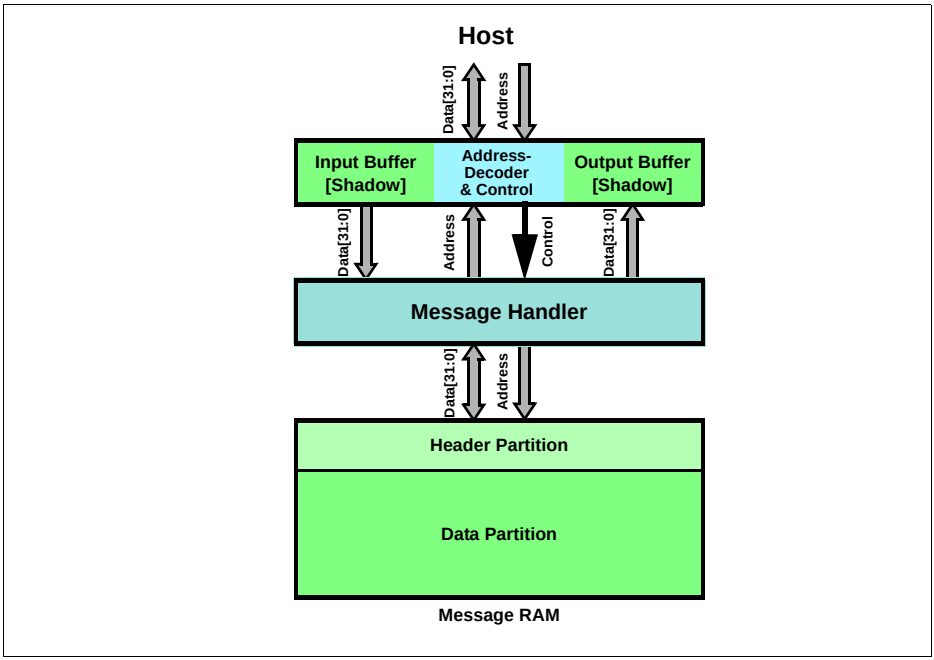


Figure 23-11 Host Access to Message RAM

Data Transfer from Input Buffer to Message RAM

To configure / update a Message Buffer in the Message RAM, the Host has to write the data to **WRDSnn (nn = 01-64)** and the Header to **WRHS1**, **WRHS2**, **WRHS3**. Two sets of **WRDSnn (nn = 01-64)** are available in parallel and selected by **CUST1.IBF1PAG**and **CUST1.IBF2PAG**. **CUST1.IBFS** shows which Input Buffer is currently used as Input Shadow Buffer and which as Input Host Buffer. **WRHS1**, **WRHS2**, and **WRHS2** does only exist once. The specific action is selected by configuring the Input Buffer Command Mask **IBCM**.

When the Host writes the number of the target Message Buffer in the Message RAM to IBRH in the Input Buffer Command Request register **IBCR**, IBF Host and IBF Shadow are swapped (see **Figure 23-12**).

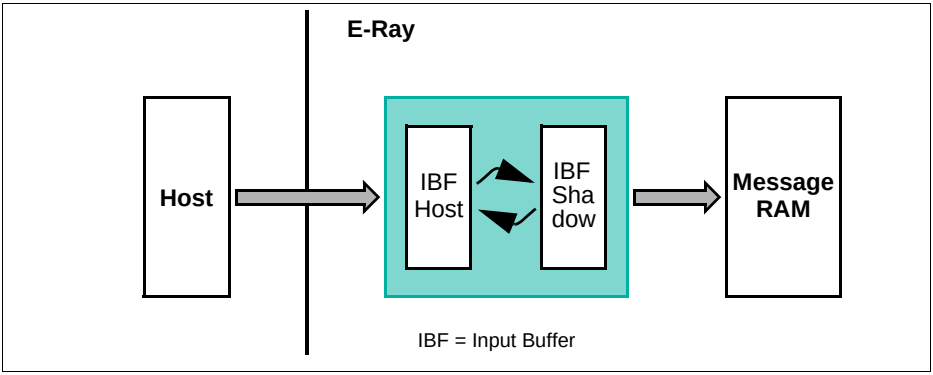


Figure 23-12 Double Buffer Structure Input Buffer

In addition the bits in the Input Buffer Command Mask and Input Buffer Command Request registers are also swapped to keep them attached to the respective IBF section (see **Figure 23-13**).

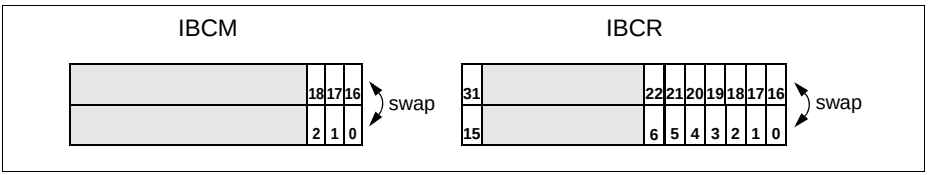


Figure 23-13 Swapping of IBCM and IBCR Bit

With this write operation the IBSYS bit in the Input Buffer Command Request register is set to 1. The Message Handler then starts to transfer the contents of IBF Shadow to the Message Buffer in the Message RAM selected by IBRS.

While the Message Handler transfers the data from IBF Shadow to the target Message Buffer in the Message RAM, the Host may write the next message to IBF Host. After the transfer between IBF Shadow and the Message RAM has completed, the IBSYS bit is set back to 0 and the next transfer to the Message RAM may be started by the Host by writing the respective target Message Buffer number to IBRH in the Input Buffer Command Request register.

If a write access to IBRH occurs while IBSYS is 1, IBSYH is set to 1. After completion of the ongoing data transfer from IBF Shadow to the Message RAM, IBF Host and IBF Shadow are swapped, IBSYH is reset to 0, IBSYS remains set to 1, and the next transfer

to the Message RAM is started. In addition the Message Buffer numbers stored under IBRH and IBRS and the Command Mask flags are also swapped.

Table 23-16 Assignment of Input Buffer Command Mask Bit

Pos.	Access	Bit	Function
18	rh	**STXRS**	Set Transmission Request Shadow
17	rh	**LDSS**	Load Data Section Shadow
16	rh	**LHSS**	Load Header Section Shadow
2	rw	**STXRH**	Set Transmission Request Host
1	rw	**LDSH**	Load Data Section Host
0	rw	**LHSH**	Load Header Section Host

Table 23-17 Assignment of Input Buffer Command Request Bit

Pos.	Access	Bit	Function
31	rh	IBSYS	**IBF Busy Shadow,** signals ongoing transfer from IBF Shadow to Message RAM
21–16	rh	IBRS	**IBF Request Shadow,** number of Message Buffer currently / last updated
15	rh	IBSYH	**IBF Busy Host,** transfer request pending for Message Buffer referenced by IBRH
5-0	rwh	IBRH	**IBF Request Host,** number of Message Buffer to be updated next

Data Transfer from Message RAM to Output Buffer

To read a Message Buffer from the Message RAM, the Host has to write to Command Request register **OBCR** to trigger the data transfer as configured in Output Buffer Command Mask **OBCM** register. After the transfer has completed, the Host can read the transferred data from **RDDSnn (nn = 01-64)**, **RDHS1**, **RDHS2**, **RDHS2**, and **MBS**.

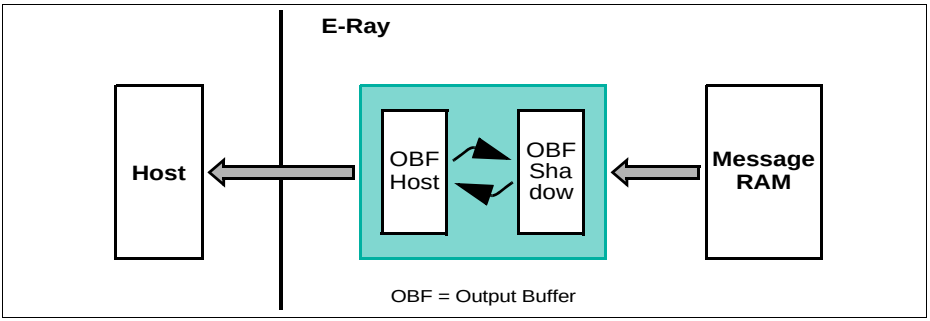


Figure 23-14 Double Buffer Structure Output Buffer

OBF Host and OBF Shadow as well as bits **OBCM.RHSS**, **OBCM.RDSS**, **OBCM.RHSH**, **OBCM.RDSH** and bits **OBCR.OBRS**, **OBCR.OBRH** are swapped under control of bits **OBCR.VIEW** and **OBCR.REQ**.

Writing bit **OBCR.REQ** to 1 copies bits **OBCM.RHSS**, **OBCM.RDSS** and bits **OBCR.OBRS** to an internal storage (see **Figure 23-15**).

After setting **OBCR.REQ** to 1, **OBCR.OBSYS** is set to 1, and the transfer of the Message Buffer selected by **OBCR.OBRS** from the Message RAM to OBF Shadow is started. After the transfer between the Message RAM and OBF Shadow has completed, the **OBCR.OBSYS** bit is set back to 0. Bits **OBCR.REQ** and **OBCR.VIEW** can only be set to 1 while **OBCR.OBSYS** is 0.

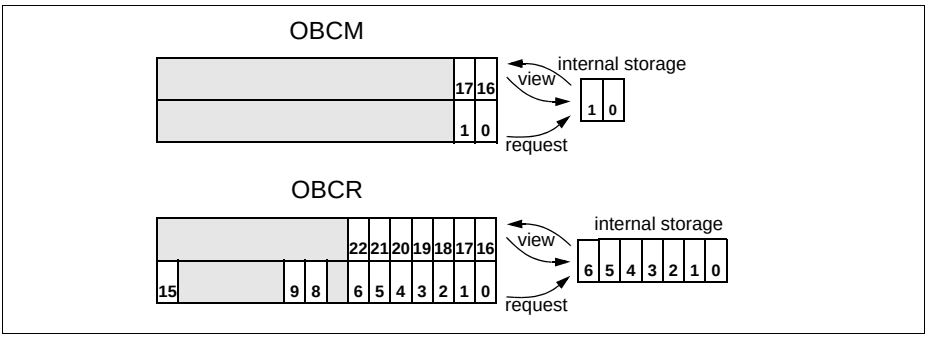


Figure 23-15 Swapping of OBCM and OBCR Bit

OBF Host and OBF Shadow are swapped by setting bit **OBCR.VIEW** to 1 while bit **OBCR.OBSYS** is 0 (see **Figure 23-14**).

In addition bits **OBCR.OBRH** and bits **OBCM.RHSH**, **OBCM.RDSH** are swapped with the registers internal storage thus assuring that the Message Buffer number stored in

OBCR.OBRH and the mask configuration stored in **OBCM.RHSH**, **OBCM.RDSH** matches the transferred data stored in OBF Host (see **Figure 23-15**).

Now the Host can read the transferred Message Buffer from OBF Host while the Message Handler may transfer the next message from the Message RAM to OBF Shadow.

Table 23-18 Assignment of Output Buffer Command Mask Bit

Pos.	Access	Bit	Function
17	rh	RDSH	Data Section available for Host access
16	rh	RHSH	Header Section available for Host access
1	rw	RDSS	Read Data Section Shadow
0	rw	RHSS	Read Header Section Shadow

Table 23-19 Assignment of Output Buffer Command Request Bit

Pos.	Access	Bit	Function
22–16	rh	OBRH	**OBF Request Host,** number of Message Buffer available for Host access
15	rh	OBSYS	**OBF Busy Shadow,** signals ongoing transfer from Message RAM to OBF Shadow
9	rw	REQ	**Request Transfer from Message RAM to OBF Shadow**
8	rwh	VIEW	**View OBF Shadow, swap OBF Shadow, and OBF Host**
6–0	rwh	OBRS	**OBF Request Shadow,** number of Message Buffer for next request

23.6.11.2 Data Transfers between IBF / OBF and Message RAM

This document uses the following terms and abbreviations:

Table 23-20 Terms and Abbreviations

Term	Meaning
MHD	Message Handler
IBF	Input Buffer 1 or 2 RAM
OBF	Output Buffer 1 or 2 RAM
MBF	Message Buffer RAM
TBF	Transient Buffer RAM Channel A (TBF1) or Channel B (TBF2)

Table 23-20 Terms and Abbreviations (cont'd)

IBF ⇒ MBF	Transfer from IBF to MBF
MBF ⇒ OBF	Transfer from MBF to OBF
MBF ⇒ TBF	Transfer from MBF to TBF
TBF ⇒ MBF	Transfer from TBF to MBF
SS	Slot Status
SS ⇒ MBF	Transfer SS to MBF

Message Handler functionality

The MHD controls the access to the MBF. It manages data-transfer between MBF and IBF, OBF, TBF1, TBF2. The data-path are shown in Figure 23-16.

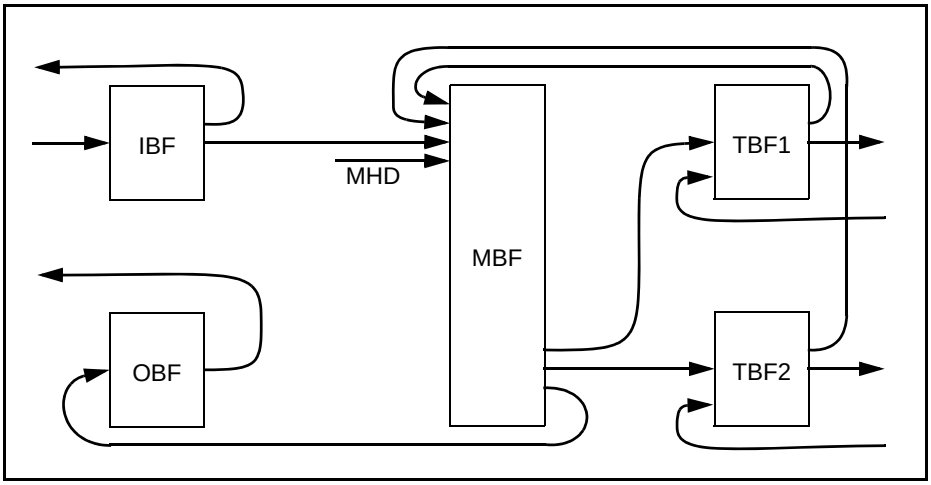


Figure 23-16 Interconnection of RAMs

Furthermore a search-algorithm allows to find the next valid message object in the MBF for transmission or reception.

Each transfer consists of a setup-time, four time steps to transfer the Header-section and a payload-length-dependent number of time steps to transfer the data-section. The internal data-busses have a width of 32 bits. Thereby it is possible to transfer two 2-byte words in one time step. If the payload consists of an odd number of 2-byte words the last time step of the data-section contains only 16 bit of valid data. If the Payload-Length (PL) is e.g. 7, the data-section consists of 4 time steps.

The maximum length for the data-section is 64 time steps, the minimum length is zero time steps.

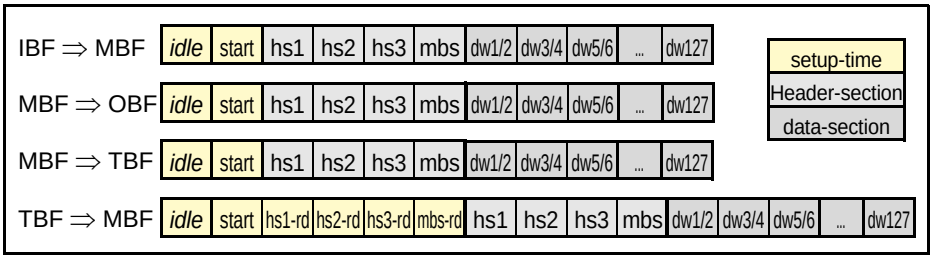


Figure 23-17 Different Possible Buffer Transfers

The update of the Slot-Status consists of a setup-time and one time-step to write the new Slot-Status.

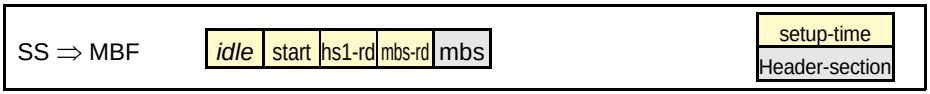


Figure 23-18 Update of Slot Status

The length of a time step depends on the number of concurrent tasks.

The following concurrent tasks are executed under control of the Message Handler:

- Data transfer between IBF or OBF and MBF
- Data transfer between TBF1 and MBF, search next TX / RX Message Buffer CHA
- Data transfer between TBF2 and MBF, search next TX / RX Message Buffer CHB

Thereby the time step length can vary between one and three f_{CLC_ERAY}periods.

Under certain conditions it is possible that a transfer is stopped or interrupted for a number of time steps until it is continued.

When a IB F⇒ MBF is started short after a TBF ⇒ MBF or SS ⇒ MBF the transfer from IBF has to wait until the setup-time of the internal transfer has finished (see Figure 23-19)

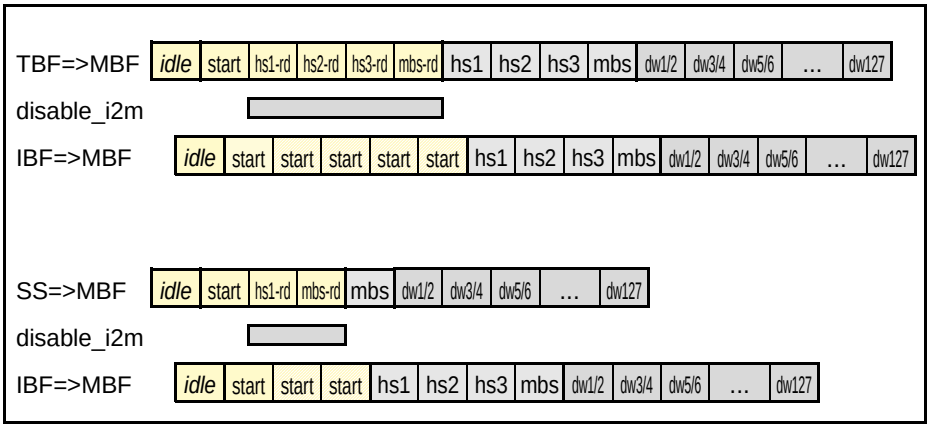


Figure 23-19 Delay start of IBF⇒MBF

The internal signal “disable_i2m” is always active when the TBF ⇒ MBF is in state “hs1-rd”, “hs2-rd”, “hs3-rd” or “mbs-rd” and when the SS ⇒ MBF is in state “hs1-rd” or “mbs-rd”.

The IBF ⇒ MBF is hold in state “start” until the internal signal “disable_i2m” gets inactive.

These additional time-steps are independent of any address-counter-values. This means, the IBF ⇒ MBF has to wait even if it writes to another buffer than the internal transfer.

Multiple requests of transfers between IBF/OBF and Message RAM

The time required to transfer the contents of a Message Buffer between IBF / OBF and Message RAM depends on the number of 4-byte words to be transferred, the number of concurrent tasks to be managed by the Message Handler, and in special cases the type and address range of the internal transfer. The number of 4-byte words varies from 4 (Header Section only) to 68 (Header + maximum Data Section) plus a short setup time to start the first transfer, while the number of concurrent task varies from one to three. The 4 Header words have to be included in calculation even if only the Data Section is requested for transfer.

The following concurrent tasks are executed under control of the Message Handler:

- Data transfer between IBF or OBF and MBF
- Data transfer between TBF1 and MBF, search next TX / RX Message Buffer CHA
- Data transfer between TBF2 and MBF, search next TX / RX Message Buffer CHB

Transfers between IBF and MBF respectively MBF and OBF can only be handled one after another. In case that e.g. a IBF ⇒ MBF has been started shortly before a

MBF ⇒ OBF is requested, the MBF ⇒ OBF has to wait until the IBF ⇒ MBF has completed.

In case that e.g. a second IBF⇒MBF is requested, a MBF⇒OBF is requested and a IBF⇒MBF is ongoing, the MBF⇒OBF has to wait until the first IBF⇒IBF has completed. The second IBF=MBF has to wait until the MBF⇒OBF has completed (see figure 23-20) independent whether MBF⇒OBF or second IBF⇒MBF is requested first.

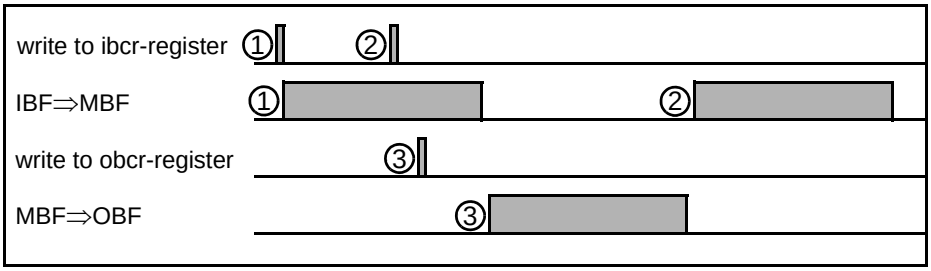


Figure 23-20 Multiple IBF/OBF Request

Worst case for single request

When a message with a large payload length is received the TBF⇒MBF is started at the begin of the next slot (n+1). If the next slot is a dynamic slot without transmission/reception (minislot), it may happen that the TBF⇒MBF has not finished until begin of the next but one slot (n+2). In this case the TBF⇒MBF will be service requested (break) to start a transmission in the next but one slot (MBF⇒TBF) and/or to update the slot status (SS⇒MBF) for the RX-buffer corresponding with next slot (n+1). After this interruption the TBF⇒MBF is continued.

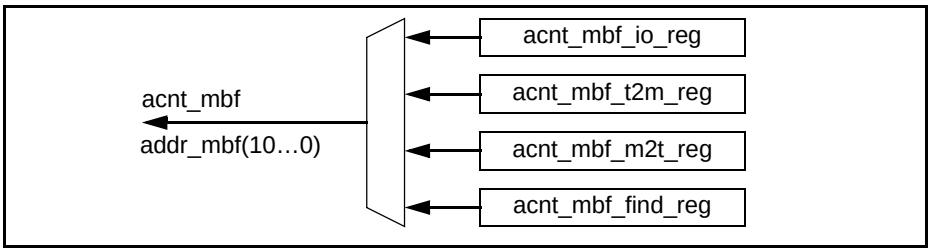


Figure 23-21 Address Counter Scheme of Message RAM (simplified)

For the transfers IBF⇒MBF / MBF⇒OBF, TBF⇒MBF and MBF⇒TBF separate address-counter are implemented (see Figure 23-21).

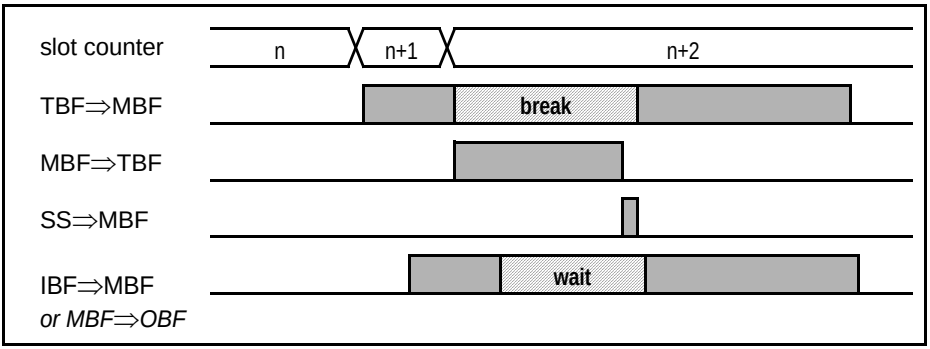


Figure 23-22 interruption of TBF⇒MBF

If the address-counter for IBF⇒MBF / MBF⇒OBF (acnt_mbf_io_reg) reaches the address of the interrupted TBF=>MBF (acnt_mbf_t2m_reg) the IBF⇒MBF / MBF⇒OBF has to wait until the TBF⇒MBF is continued (see Figure 23-22).

The relative time is measured in f_{CLC_ERAY} cycles. Absolute time depends on the actual f_{CLC_ERAY} cycle period.

$$\text{tbf_to_mbf_break time}_{max} = (\text{setup time} + \text{mbf_to_tbf time}_{max}) + (\text{setup time} + \text{ss_to_mbf})$$

$$\text{cycles}_{req} = (\text{number of concurrent tasks}) \times ((\text{setup time} + (\text{number of 4-byte words})_{req}) + \text{tbf_to_mbf_break time})$$

$$\text{setup time} = 2\ f_{CLC_ERAY}\ \text{cycles}$$

Worst case for one IBF⇒MBF or MBF⇒OBF:

Max. break time: tbf_to_mbf_break time$_{max}$ = (2+68) + (4+1) = 75

Max. number of f_{CLC_ERAY} cycles: cycles$_{req}$ = 3 × (6 + 68 + 75) = 435

Worst case for multiple transfers

If a second IBF⇒MBF and a MBF⇒OBF (see Figure 23-20) is requested directly after the first IBF⇒MBF has started following worst case timing could appear:

cycles$_{trans}$ = (remaining cycles of transfer running)
+ (cycles of second requested transfer)
+ (cycles of third requested transfer)

cycles$_{trans}$ = cycles$_{rem}$ + cycles$_{req_2}$+ cycles$_{req_3}$

Max. number of f_{CLC_ERAY} cycles: cycles$_{trans}$ = 447 + 435 + 447 = 1329

23.6.11.3 Minimum f_{CLC_ERAY}

To calculate the minimum f_{CLC_ERAY} the worst case scenario has to be considered.

The worst case scenario depends on the following parameters

- maximum payload length
- minimum minislot length
- number of configured Message Buffers (excluding FIFO)
- used channels (single/dual channel)

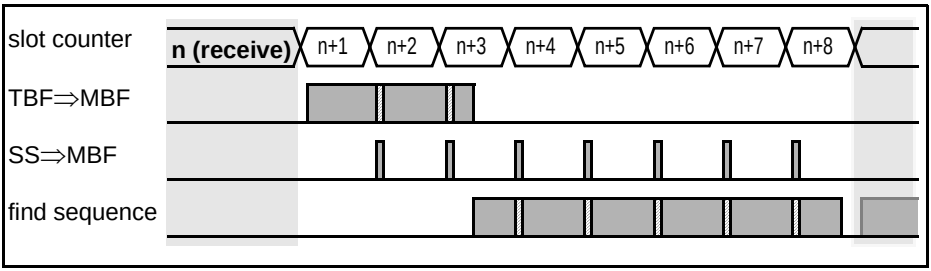


Figure 23-23 worst case scenario

Worst case scenario:

- reception of message with a maximum payload length in Slot n (n is 7,15,23,31,39,…)
- slot n+1 to n+7 are empty dynamic slots (minislot) and configured as receive buffer
- the find-sequence (usually started in slot 8,16,24,32,40,…) has to scan the maximum number of configured buffers
- the number of concurrent tasks has its maximum value of three

The find-sequence is executed each 8 Slots (slot 8,16,24,32,40,…). It has to be finished until the next find-sequence is requested.

The length of a TBF⇒MBF varies from 4 (Header Section only) to 68 (Header + maximum Data Section) time step plus a setup time of 6 time steps.

$$cycles_{t2m} = \frac{f_{CLC_ERAY}}{\text{number of concurrent tasks}) \times (\text{setup time}_{t2m} + (\text{number of 4-byte words})_{t2m})}$$

A SS⇒MBF has a fixed length of 1 time steps plus a setup time of 4 time steps.

$$cycles_{ss2m} = \frac{f_{CLC_ERAY}}{(\text{number of concurrent tasks}) \times 5}$$

The find sequence has a maximum length of 128 (maximum number of buffers) time steps plus a setup time of 2 time steps.

$$cycles_{find} = \frac{f_{CLC_ERAY}}{(\text{number of concurrent tasks}) \times (\text{setup time}_{find} + (\text{number of configured buffers}))}$$

A minislot has a length of 2 to 63 Macrotick (gdMinislot). The minimum nominal Macrotick period (cdMinMTNom) is 1 µs. A sequence of 8 minislots has a length of

$$time_{8minislots} = 8 \times gdMinislot \times cdMinMTNom$$

The maximum period $T_{CLC_ERAY} = 1/f_{CLC_ERAY}$ can be calculated as followed:

$$time_{8minislots} \geq (f_{CLC_ERAY}\ \text{period in}\ \mu s) \times (\ (f_{CLC_ERAY}\ cycles_{t2m}) + 7 \times (f_{CLC_ERAY}\ cycles_{ss2m}) + (f_{CLC_ERAY}\ cycles_{find}))$$

$$f_{CLC_ERAY}\ \text{period in ms} \leq \frac{time_{8mnislots}}{(\ cycles_{t2m}) + 7 \times (cycles_{ss2m}) + (cycles_{find})}$$

minimum $time_{8minislots}$ = $8 \times 2 \times 1\ \mu s = 16\ \mu s$

maximum $f_{CLC_ERAY}\ cycles_{t2m}$ = $3 \times (6 + 68) = 222$

maximum $f_{CLC_ERAY}\ cycles_{ss2m}$ = $3 * 5 = 15$

maximum $f_{CLC_ERAY}\ cycles_{find}$ = $3 * (2 + 128) = 390$

$$f_{CLC_ERAY}\ \text{period in ms} \leq \frac{16\mu s}{222 + 7 \times 15 + 390} = 22.315\ldots ns$$

The minimum f_{CLC_ERAY} frequency for this worst case scenario is 44.8125 MHz.

A too low f_{CLC_ERAY} frequency can cause a malfunction of the E-Ray.

The E-Ray can detect several malfunctions and reports this by setting the corresponding flag in the Message Handler Constraints Flags (**MHDF**) register.

Minimum f_{CLC_ERAY} for various maximum payload length

Table 23-21 summarizes the minimum required f_{CLC_ERAY} frequency for various maximum payload length assuming:

- a minimum minislot length of 2µs.
- a maximum of 128 configured Message Buffers.
- dual channels in use.

Table 23-21 Minimum f_{CLC_ERAY} for different maximum payload length

Maximum payload length of 32 bit words	4	8	16	32	64
minimum f_{CLC_ERAY}	32,82 MHz	33,57 MHz	35,07 MHz	38,07 MHz	44,1 MHz

Minimum f_{CLC_ERAY} for various minimum minislot length

Table 23-22 summarizes the minimum required f_{CLC_ERAY} frequency for various minimum minislot length assuming:

- a maximum payload length of 254 bytes / 64 four-byte-words.
- a maximum 128 configured Message Buffers.
- dual channels in use.

Table 23-22 Minimum f_{CLC_ERAY} for different minimum minislot length

gdMinislot at dMinMTNom = 1 µs	2 µs	3 µs	4 µs	7 µs	8 µs
minimum f_{CLC_ERAY}	44,82 MHz	29,88 MHz	22,412 MHz	12,8 MHz	9,96 MHz

Minimum f_{CLC_ERAY} for various amount of configured Message Buffers

Table 23-23 summarizes the minimum required f_{CLC_ERAY} frequency for various amount of configured Message Buffers assuming:

- a maximum payload length of 254 bytes / 64 four-byte-words.
- a minimum minislot length of 2 µs.
- dual channels in use.

Table 23-23 Minimum f_{CLC_ERAY} for different amount of configured Message Buffers

Configured maximum amount of Message Buffers	128	64	32
minimum f_{CLC_ERAY}	44,82 MHz	32,82 MHz	26,82 MHz

Minimum f_{CLC_ERAY} for a typical configuration

´The minimum required f_{CLC_ERAY} frequency for various assuming the following typical E-Ray configuration:

- a maximum payload length of 32 bytes / 8 four-byte-words.
- a minimum minislot length of 7 µs.
- a maximum 128 configured Message Buffers.
- dual channels in use

The minimum f_{CLC_ERAY} frequency for this typical example would be 10 MHz.

23.6.11.4 FlexRay™ Protocol Controller access to Message RAM

The two Transient Buffer RAMs (TBF 1, TBF 2) are used to buffer the data for transfer between the two FlexRay™ Protocol Controllers and the Message RAM.

Each Transient Buffer RAM is build up as a double buffer, able to store two complete FlexRay™ messages. There is always one buffer assigned to the corresponding Protocol Controller while the other one is accessible by the Message Handler.

If e.g. the Message Handler writes the next message to be send to Transient Buffer Tx, the FlexRay™ Channel Protocol Controller can access Transient Buffer Rx to store the message it is actually receiving. During transmission of the message stored in Transient Buffer Tx, the Message Handler transfers the last received message stored in Transient Buffer Rx to the Message RAM (if it passes acceptance filtering) and updates the respective Message Buffer.

Data transfers between the Transient Buffer RAMs and the shift registers of the FlexRay™ Channel Protocol Controllers are done in words of 32 bit. This enables the use of a 32 bit shift register independent of the length of the FlexRay™ messages.

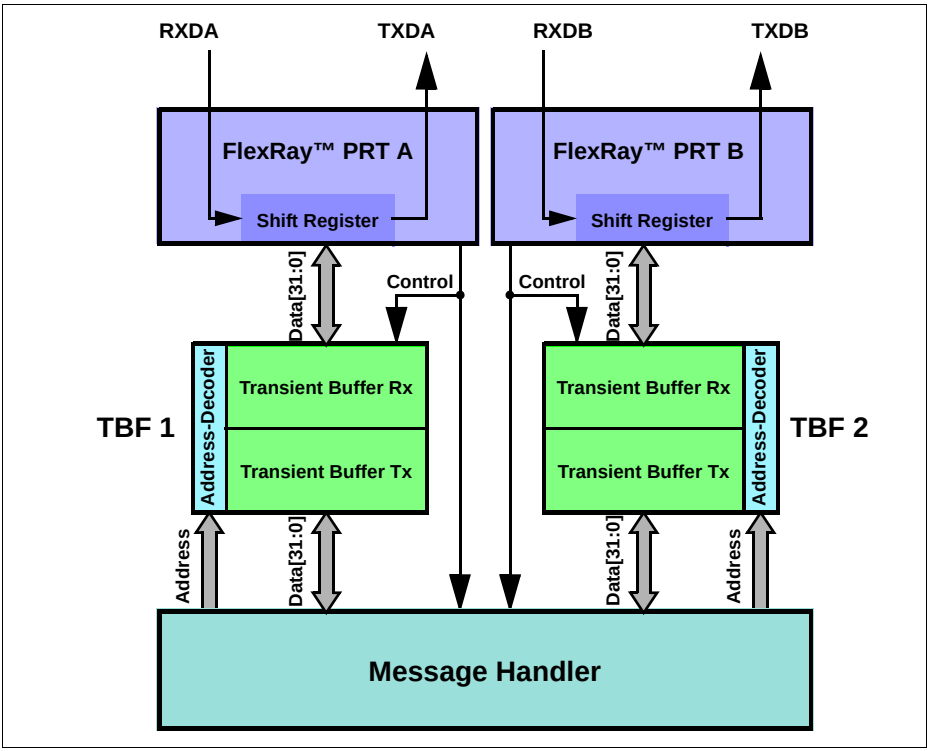


Figure 23-24 Access to Transient Buffer RAMs

23.6.12 Message RAM

To avoid conflicts between Host access to the Message RAM and FlexRay™ message reception / transmission, the Host cannot directly access the Message Buffers in the Message RAM. These accesses are handled via the Input and Output Buffers. The Message RAM is able to store up to 128 Message Buffers depending on the configured payload length.

The Message RAM is organized 2048 x 32. To achieve the required flexibility with respect to different numbers of data byte per FlexRay™ Frame (0 to 254), the Message RAM has a structure as shown in **Figure 23-25**.

The Data Partition is allowed to start at Message RAM word number: (MRC.LCB + 1) • 4

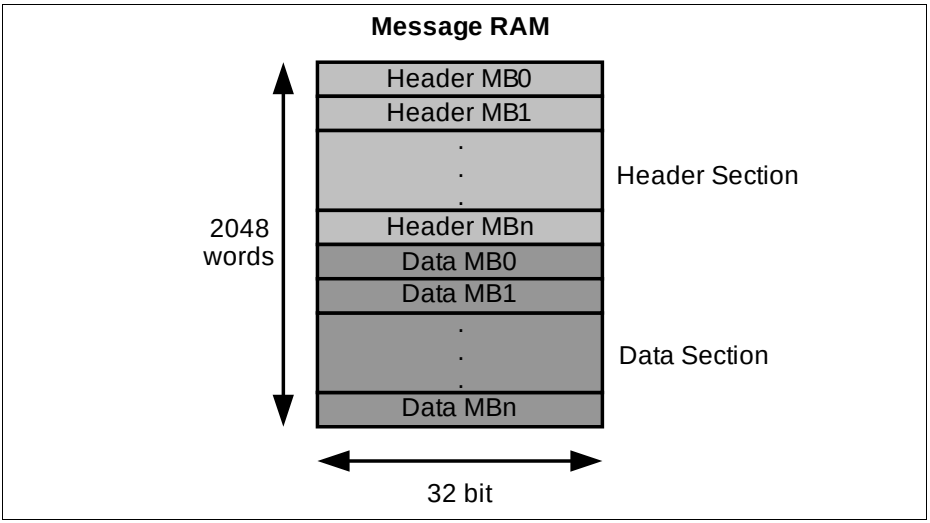


Figure 23-25 Structure of Message RAM

Header Partition

Stores Header Segments of FlexRay™ Frames:

- Supports a maximum of 128 Message Buffers
- Each Message Buffer has a Header of four 32 bit words
- Header 3 of each Message Buffer holds the11 bit pointer to the respective Data Section in the Data Partition

Data Partition

Flexible storage of Data Sections with different length. Some maximum values are:

- 30 Message Buffers with 254 byte Data Section each
- Or 56 Message Buffers with 128 byte Data Section each
- Or 128 Message Buffers with 48 byte Data Section each

Restriction: Header Partition + Data Partition may not occupy more than 2048 32-bit words.

23.6.12.1 Header Partition

The Header of each Message Buffer occupies four 32-bit words in the Header Partition of the Message RAM. The Header of Message Buffer 0 starts with the first word in the Message RAM.

For transmit buffers the Header CRC has to be calculated by the Host.

Payload Length Received **PLR**, Receive Cycle Count **RCC**, Received on Channel Indication **RCI**, Startup Frame Indication bit **SFI**, Sync bit **SYN**, NULL Frame Indication bit **NFI**, Payload Preamble Indication bit **PPI**, and Reserved bit **RES** are only updated from received valid Frames (including valid NULL Frames).

Header word 4 of each configured Message Buffer holds the respective Message Buffer Status **MBS** information.

Table 23-24 Header Section of a Message Buffer in the Message RAM

Bit Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1			MBI	TXM	NME	CFG	CHB	CHA		Cycle Code												Frame ID										
2		Payload Length Received								Payload Length Configured												Tx Buffer: Header CRC Configured Rx Buffer: Header CRC Received										
3			RES	PPI	NFI	SYN	SFI	RCI		Receive Cycle Count												Data Pointer										
4			RESS	PPIS	NFIS	SYNS	SFIS	RCIS		Cycle Count Status							TFB	FTY		MLST	ESB	ESA	TCIB	TCIA	SVOB	SVOA	CEOB	CEOA	SEOB	SEOA	VFRB	VFRA

- Frame Configuration
- Filter Configuration
- Message Buffer Control
- Message RAM Configuration
- Updated from received Frame
- Message Buffer Status
- unused

Header 1 (word 0)

Write access via **WRHS1**, read access via **RDHS1**:

- Frame ID: Slot counter filtering configuration
- Cycle Code: Cycle counter filtering configuration
- CHA, CHB: Channel filtering configuration
- CFG: Message Buffer configuration: receive / transmit
- PPIT: Payload Preamble Indicator Transmit
- XMI: Transmit mode configuration: single-shot / continuous
- MBI: Message Buffer receive / transmit service request enable

Header 2 (word 1)

Write access via **WRHS2**, read access via **RDHS2**:

- Header CRC
  - Transmit Buffer: Configured by the Host (calculated from Frame Header Segment)
  - Receive Buffer: Updated from received Frame
- Payload Length Configured
  - Length of Data Section (2-byte words) as configured by the Host
- Payload Length Received
  - Length of Payload Segment (2-byte words) stored from received Frame

Header 3

Write access via **WRHS3**, read access via **RDHS3**:

- Data Pointer
  - Pointer to the beginning of the corresponding Data Section in the Data Partition

Read access via **RDHS3**, valid for receive buffers only, updated from received Frames:

- Receive Cycle Count: Cycle count from received Frame
- RCI: Received on Channel Indicator
- SFI: Startup Frame Indicator
- SYN: SYNC Frame Indicator
- NFI: NULL Frame Indicator
- PPI: Payload Preamble Indicator
- RES: Reserved bit

Message Buffer Status MBS (word 3)

Read access via MBS, updated by the Communication Controller at the end of the configured slot.

- VFRA: Valid Frame Received on channel A
- VFRB: Valid Frame Received on channel B
- SEOA: Syntax Error Observed on channel A
- SEOB: Syntax Error Observed on channel B
- CEOA: Content Error Observed on channel A
- CEOB: Content Error Observed on channel B
- SVOA: Slot boundary Violation Observed on channel A
- SVOB: Slot boundary Violation Observed on channel B
- TCIA: Transmission Conflict Indication channel A
- TCIB: Transmission Conflict Indication channel B
- ESA: Empty Slot Channel A
- ESB: Empty Slot Channel B
- MLST: Message LoST
- FTA: Frame Transmitted on Channel A
- FTA: Frame Transmitted on Channel B
- Cycle Count Status: Actual cycle count when status was updated
- RCIS: Received on CHannel Indicator Status
- SFIS: Startup Frame Indicator Status
- SYNS: SYNC Frame Indicator Status
- NFIS: NULL Frame Indicator Status
- PPIS: Payload Preamble Indicator Status
- RESS: Reserved Bit Status

23.6.12.2 Data Partition

The Data Partition of the Message RAM stores the Data Sections of the Message Buffers configured for reception / transmission as defined in the Header Partition. The number of data bytes for each Message Buffer can vary from 0 to 254. To optimize the data transfer between the shift registers of the two FlexRay™ Protocol Controllers and the Message RAM as well as between the Host interface and the Message RAM, the physical width of the Message RAM is set to 4 bytes.

The Data Partition starts after the last word of the Header Partition. When configuring the Message Buffers in the Message RAM the programmer has to assure that the data pointers point to addresses within the Data Partition. **Table 23-25** below shows an example how the Data Sections of the configured Message Buffers can be stored in the Data Partition of the Message RAM.

The beginning and the end of a Message Buffer's Data Section is determined by the data pointer and the payload length configured in the Message Buffer's Header Section,

respectively. This enables a flexible usage of the available RAM space for storage of Message Buffers with different data length.

If the size of the Data Section is an odd number of 2-byte words, the remaining 16 bits in the last 32-bit word are unused (see **Table 23-25** below)

Table 23-25 Example for Structure of the Data Section in the Message RAM

Bit Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
…	unused								unused								unused								unused							
…	unused								unused								unused								unused							
…	**MB1** Data3								**MB1** Data2								**MB1** Data1								**MB1** Data0							
…	…								…								…								…							
…	…								…								…								…							
…	**MB1** Data(n)								**MB1** Data(n-1)								**MB1** Data(n-2)								**MB1** Data(n-3)							
…	…								…								…								…							
…	…								…								…								…							
…	…								…								…								…							
…	**MB1** Data3								**MB1** Data2								**MB1** Data1								**MB1** Data0							
…	…								…								…								…							
…	**MB1** Data(k)0								**MB1** Data(k-1)0								**MB1** Data(k-2)0								**MB1** Data(k-3)0							
2046	**MB80** Data3								**MB80** Data2								**MB80** Data2								**MB80** Data0							
2047	unused								unused								**MB80** Data5								**MB80** Data4							

23.6.12.3 ECC Check

There is an ECC checking mechanism implemented in the E-Ray module to assure the integrity of the data stored in the seven RAM blocks of the module. The RAM blocks have an ECC generator / checker attached as shown in **Figure 23-26**. When data is written to a RAM block, the local ECC generator generates the ECC data. The ECC data is stored together with the respective data word. The ECC data is checked each time a data word is read from any of the RAM blocks.

If an ECC error is detected, the respective error flag is set. The ECC error flags **MHDS.EIBF**, **MHDS.EOBF**, **MHDS.EMR**, **MHDS.ETBF1**, **MHDS.ETBF2**, and the faulty Message Buffer indicators **MHDS.FMBD**, **MHDS.MFMB**, **MHDS.FMB** are located in the Message Handler Status register. These error flags control the error interrupt flag **EIR.EERR**.

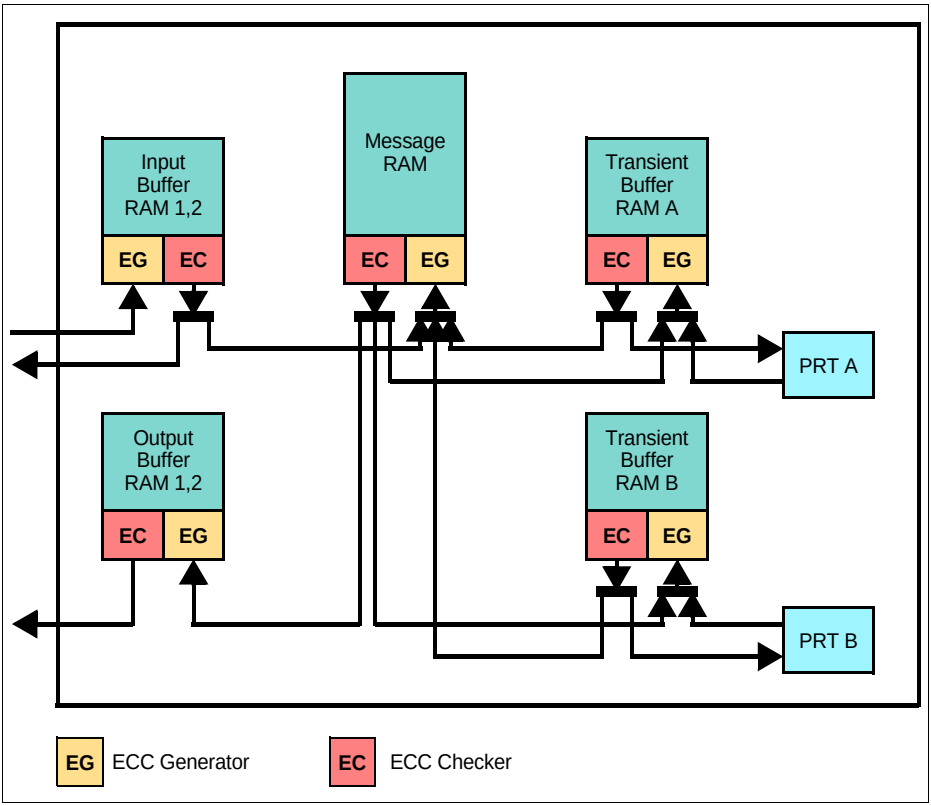


Figure 23-26 ECC Generation and Check

When an ECC error has been detected the following actions will be performed:

In all cases

- The respective ECC error flag in the Message Handler Status **MHDS** register is set
- The ECC error flag **EIR.EERR** in the Error Service Request Register is set, and if enabled, a module service request to the Host will be generated.

Additionally in specific cases

1. ECC error in data transfer from Input Buffer RAM 1,2 ⇒ Message RAM (Transfer of Header and Data Section)
   a) **MHDS.EIBF** bit is set
   b) **MHDS.FMBD** bit is set to indicate that **MHDS.FMB** has been updated
   c) **MHDS.FMB** indicates the number of the faulty Message Buffer
   d) Transmit buffer: Transmission request for the respective Message Buffer is not set
2. ECC error in data transfer from Input Buffer RAM 1,2 ⇒ Message RAM (Transfer of Data Section only)
   a) **MHDS.EMR** bit is set
   b) **MHDS.FMBD** bit is set to indicate that **MHDS.FMB** points to a faulty Message Buffer
   c) **MHDS.FMB** indicates the number of the faulty Message Buffer
   d) The Data Section of the respective Message Buffer is not updated
   e) Transmit buffer: Transmission request for the respective Message Buffer is not set
3. ECC error during host reading Input Buffer RAM
   a) • **MHDS.EIBF** bit is set
4. ECC error during scan of Header Sections in Message RAM
   a) **MHDS.EMR** bit is set
   b) **MHDS.FMBD** bit is set to indicate that **MHDS.FMB** points to a faulty Message Buffer
   c) **MHDS.FMB** indicates the number of the faulty Message Buffer
   d) Ignore Message Buffer (Message Buffer is skipped)
5. ECC error during data transfer from Message RAM ⇒ Transient Buffer RAM A, B
   a) **MHDS.EMR** bit is set
   b) **MHDS.FMBD** bit is set to indicate that **MHDS.FMB** points to a faulty Message Buffer
   c) **MHDS.FMB** indicates the number of the faulty Message Buffer
   d) Frame not transmitted, Frames already in transmission are invalidated by setting the Frame CRC to zero
6. ECC error during data transfer from Transient Buffer RAM A, B ⇒ Protocol Controller 1, 2
   a) **MHDS.ETBF1**, **MHDS.ETBF2** bit is set
7. ECC error in data transfer from Transient Buffer RAM A, B ⇒ Message RAM (ECC error when reading Header Section of respective Message Buffer from Message RAM)
   a) **MHDS.EMR** bit is set
   b) **MHDS.FMBD** bit is set to indicate that **MHDS.FMB** points to a faulty Message Buffer
   c) **MHDS.FMB** indicates the number of the faulty Message Buffer
   d) The Data Section of the respective Message Buffer is not updated

8. ECC error in data transfer from Transient Buffer RAM A, B ⇒ Message RAM (ECC error when reading Transient Buffer RAM A, B)
   a) **MHDS.ETBF1**, **MHDS.ETBF2** bit is set
   b) **MHDS.FMBD** bit is set to indicate that **MHDS.FMB** points to a faulty Message Buffer
   c) **MHDS.FMB** indicates the number of the faulty Message Buffer
9. ECC error during data transfer from Message RAM ⇒ Output Buffer RAM
   a) **MHDS.EMR** bit is set
   b) **MHDS.FMBD** bit is set to indicate that **MHDS.FMB** points to a faulty Message Buffer
   c) **MHDS.FMB** indicates the number of the faulty Message Buffer
10. ECC error during Host reading Output Buffer RAM
    a) • **MHDS.EOBF** bit is set
11. ECC error during data read of Transient Buffer RAM A, B

If an ECC error occurs while the Message Handler reads a Frame with Network Management information (PPI = 1) from the Transient Buffer RAM A, B the corresponding Network Management vector registers NMV1 to NMV3 are not updated from that Frame.

23.6.13 Host Handling of Errors

An ECC error caused by transient bit flips can be fixed by:

23.6.13.1 Self-Healing

ECC errors located in

- Input Buffer RAM 1,2
- Output Buffer RAM 1,2
- Data Section of Message RAM
- Transient Buffer RAM A
- Transient Buffer RAM B

are overwritten with the next write access to the disturbed bit(s) caused by Host access or by FlexRay communication.

23.6.13.2 CLEAR_RAMS Command

When called in DEFAULT_CONFIG or CONFIG state POC command CLEAR_RAMS initializes all module-internal RAMs to zero.

23.6.13.3 Temporary Unlocking of Header Section

An ECC error in the header section of a locked message buffer can be fixed by a transfer from the Input Buffer to the locked buffer Header Section. For this transfer, the write-

access to the IBCR (specifying the message buffer number) must be immediately preceded by the unlock sequence normally used to leave CONFIG state (see **“Lock Register (LCK)” on Page 23-30**).

For that single transfer the respective message buffer header is unlocked, regardless whether it belongs to the FIFO or whether its locking is controlled by MRC.SEC[1:0], and will be updated with new data.

23.7 Module Service Request

In general, service requests provide a close link to the protocol timing as they are triggered almost immediately when an error or status change is detected by the controller, a Frame is received or transmitted, a configured timer service request is activated, or a stop watch event occurred. This enables the Host to react very quickly on specific error conditions, status changes, or timer events. On the other hand too many service requests can cause the Host to miss deadlines required for the application. Therefore the Communication Controller supports disable / enable controls for each individual service request source separately.

An service request may be triggered when

- An error was detected
- A status flag is set
- A timer reaches a preconfigured value
- A message transfer from Input Buffer to Message RAM or from Message RAM to Output Buffer has completed
- A stop watch event occurred

Tracking status and generating service requests when a status change or an error occurs are two independent tasks. Regardless of whether an service request is enabled or not, the corresponding status is tracked and indicated by the Communication Controller. The Host has access to the actual status and error information by reading the Error Service Request Register **EIR** and the Status Service Request **SIR** Register.

Table 23-26 Module Service Request Flags and Service Request Line Enable

Register	Bit	Function
SIR	WST	Wakeup Status
	CAS	Collision Avoidance Symbol
	CYCS	Cycle Start Service Request
	TXI	Transmit Service Request
	RXI	Receive Service Request
	RFNE	Receive FIFO not Empty
	RFF	Receive FIFO Full
	NMVC	Network Management Vector Changed
	TI0	Timer Service Request 0
	TI1	Timer Service Request 1
	TIBC	Transfer Input Buffer Completed
	TOBC	Transfer Output Buffer Completed
	SWE	Stop Watch Event
	SUCS	Startup Completed Successfully
	MBSI	Message Buffer Status Interrupt
	SDS	Start of Dynamic Segment
	WUPA	Wakeup Pattern Channel A
	MTSA	MTS Received on Channel A
	WUPB	Wakeup Pattern Channel B
	MTSB	MTS Received on Channel B
ILE	EINT0	Enable Service Request Line 0
	EINT1	Enable Service Request Line 1
EIR	PEMC	Protocol Error Mode Changed
	CNA	Command Not Valid
	SFBM	SYNC Frames Below Minimum
	SFO	SYNC Frame Overflow
	CCF	Clock Correction Failure
	CCL	CHI Command Locked
	EERR	ECC Error
	RFO	Receive FIFO Overrun

Table 23-26 Module Service Request Flags and Service Request Line Enable

Register	Bit	Function
EIR	EFA	Empty FIFO Access
	IIBA	Illegal Input Buffer Access
	IOBA	Illegal Output Buffer Access
	MHF	Message Handler Constraints Flag
	EDA	Error Detected on Channel A
	LTVA	Latest Transmit Violation Channel A
	TABA	Transmission Across Boundary Channel A
	EDB	Error Detected on Channel B
	LTVB	Latest Transmit Violation Channel B
	TABB	Transmission Across Boundary Channel B

The interrupt lines to the Host TINT0SR and TINT1SR are controlled by the enabled interrupts. In addition each of the two interrupt lines can be enabled / disabled separately by programming bit ILE.EINT0/INT0SRC.SRE and ILE.EINT1/ INT1SRC.SRE.

The interrupt lines to the Host NDAT0SR and NDAT1SR are controlled by the enabled new data interrupts (**NDIC1** to **NDIC4**). In addition each of the two interrupt lines can be enabled / disabled separately by programming bit NDAT0SRC.SRE and NDAT1SRC.SRE.

The interrupt lines to the Host MBSC0SR and MBSC1SR are controlled by the enabled new data interrupts (**MSIC1** to **MSIC4**). In addition each of the two interrupt lines can be enabled / disabled separately by programming bit MBSC0SRC.SRE and MBSC1SRC.SRE.

The two timer service requests generated by service request timer 0 and 1 are available on pins TINT0SR and TINT1SR. They can be configured via the Timer 0 and Timer 1 Configuration register. In addition each of the two interrupt lines can be enabled / disabled separately by programming bit TINT0SRC.SRE and TINT1SRC.SRE.

A stop watch event may be triggered via input pin STPWn.

The status of the data transfer between IBF / OBF and the Message RAM is signalled on signals IBUSY and OBUSY. When a transfer has completed bit **SIR.TIBC** or **SIR.TOBC** is set.

23.8 Restrictions

The following restrictions have to be considered when programming the E-Ray IP-module. A violation of these restrictions may lead to an erroneous behavior of the E-Ray IP-module.

23.8.1 Message Buffers with the same Frame ID

If two or more Message Buffers are configured with the same Frame ID, and if they have a matching cycle counter filter value for the same slot, then the Message Buffer with the lowest Message Buffer number is used.

Sharing of a static time slot via cycle counter filtering between different nodes of a FlexRay™ network is **not** allowed.

23.8.2 Data Transfers between IBF / OBF and Message RAM

The time required to transfer the contents of a Message Buffer between IBF / OBF and Message RAM depends on the setup time to start the first transfer, the number of 4-byte words to be transferred, and the number of concurrent tasks to be managed by the Message Handler. The number of 4-byte words varies from 4 (Header Section only) to 68 (Header + maximum Data Section) while the number of concurrent task varies from one to three.

The following concurrent tasks are executed under control of the Message Handler:

- Data transfer between IBF or OBF and Message RAM
- Data transfer between TBF1 and Message RAM, search next TX / RX Message Buffer CHA
- Data transfer between TBF2 and Message RAM, search next TX / RX Message Buffer CHB

Transfers between IBF and Message RAM respectively Message RAM and OBF can only be handled one after another. In case that e.g. a transfer between IBF and Message RAM has been started shortly before a transfer between Message RAM and OBF is requested, the OBF transfer has to wait until the IBF transfer has completed.

The relative time is measured in f_{CLC_ERAY} cycles. Absolute time depends on the actual f_{CLC_ERAY} cycle period.

cyclestrans = (remaining cycles of transfer running) + (cycles of requested transfer)

cyclestrans = cyclesrem + cyclesreq

cyclesrem = (number of concurrent tasks) * (setup time + (number of 4-byte words)rem)

cyclesreq = (number of concurrent tasks) * (setup time + (number of 4-byte words)req)

setup time = 2 f_{CLC_ERAY} cycles

Under worst case conditions a transfer is requested directly after the previous transfer started:

Max. number of f_{CLC_ERAY} cycles: cyclestrans = (3 * (2 + 68)) + (3 * (2 + 68)) = 420

Worst case timing: timetrans(40MHz) = 420 * 25ns = 10.5 ms

23.9 E-Ray Module Implementation

This section describes the E-Ray interfaces as implemented in TC21x/TC22x/TC23x with the clock control, port and DMA connections, interrupt control, and address decoding.

Figure 23-27 shows a detailed view of the E-Ray interface.

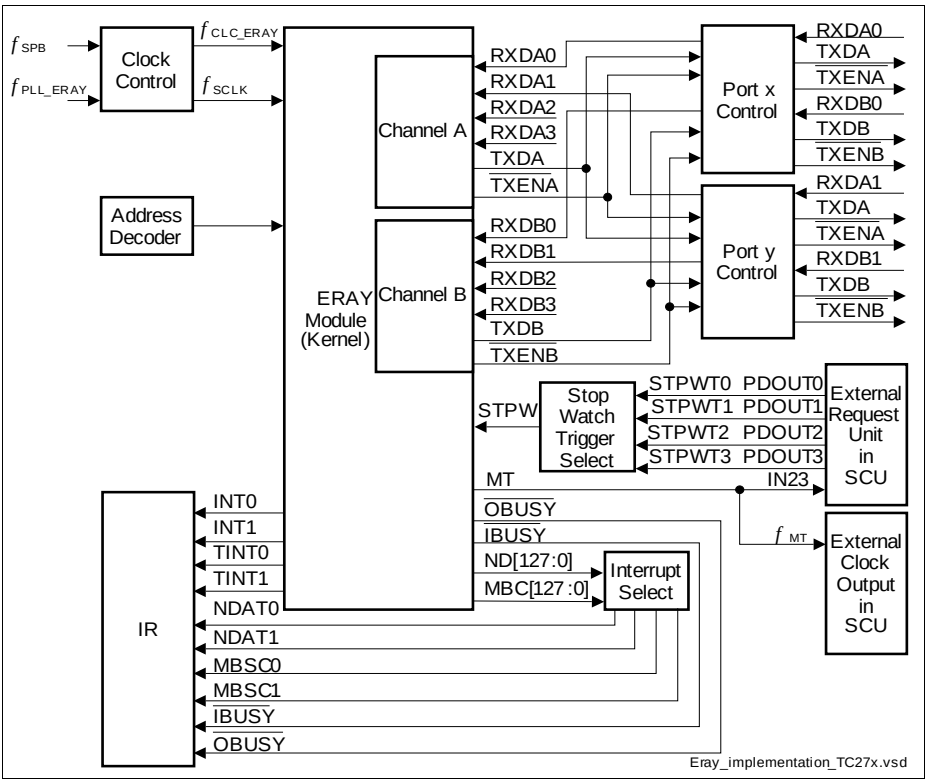


Figure 23-27 Detailed Block Diagram of the E-Ray Interface

23.9.1 Interconnections of the E-Ray Module

The E-Ray module has 2 FlexRay™ communication channels, channel A and channel B. Each channel provides a set of signals to drive a bus driver. The E-Ray module requires two different clocks, a sampling clock of the FlexRay™ bus f_{SCLK}. f_{SCLK} has to

be 8 times the baud rate of the FlexRay™ communication. A second clock f_{CLC_ERAY} is used for the main protocol controller state machine and the customer interface logic. To enable deactivation of the E-Ray Module, f_{CLC_ERAY} and f_{SCLK} may be disabled (clock gated) by the **CLC.DISR** Enable E-Ray (Clock Gating) bit.The following items are described in this section:

- E-Ray module (kernel) external registers
- Port control and connections
  - I/O port line assignment
  - I/O function selection
  - Pad driver characteristics selection
- On-chip connections
  - SCU Connections
  - DMA connections
- Module clock generation
- Interrupt registers
- E-Ray address map

23.9.2 Port Control and Connections

This section describes the I/O connections of the E-Ray module.

23.9.2.1 Input/Output Function Selection

Table 23-27 shows how bits and bit fields must be programmed for the required I/O functionality of the E-Ray I/O lines. This table also shows the values of the peripheral input select registers.

Table 23-27 E-Ray I/O Control Selection and Setup

Port Lines	Input Select Register	Input/Output Control Register Bits	I/O
FlexRay™ Channel A			
RXDA0/ P14.8	ERAY_**CUST1.RISA** = 00_B	P14_IOCR8.PC8 = $0XXXX_B$	In
RXDA1/ P11.9	ERAY_**CUST1.RISA** = 01_B	P11_IOCR8.PC9 = $0XXXX_B$	In
RXDA2/ P02.1	ERAY_**CUST1.RISA** = 10_B	P02_IOCR0.PC1 = $0XXXX_B$	In
RXDA3/ P14.1	ERAY_**CUST1.RISA** = 11_B	P14_IOCR0.PC1 = $0XXXX_B$	In
TXDA/ P02.0	not applicable	P02_IOCR0.PC0 = $1X110_B$	Out

Table 23-27 E-Ray I/O Control Selection and Setup (cont'd)

Port Lines	Input Select Register	Input/Output Control Register Bits	I/O
TXDA/ P11.3	not applicable	P11_IOCR0.PC3 = $1X100_B$	Out
TXDA/ P14.0	not applicable	P14_IOCR0.PC0 = $1X011_B$	Out
$\overline{\text{TXENA}}$/ P02.4	not applicable	P2_IOCR4.PC4 = $1X110_B$	Out
$\overline{\text{TXENA}}$/ P11.6	not applicable	P11_IOCR4.PC6 = $1X100_B$	Out
FlexRay™ Channel B			
RXDB0/ P14.7	ERAY_**CUST1.RISB** = 00_B	P14_IOCR4.PC7 = $0XXXX_B$	In
RXDB1/ P11.10	ERAY_**CUST1.RISB** = 01_B	P11_IOCR8.PC10 = $0XXXX_B$	In
RXDB2/ P02.3	ERAY_**CUST1.RISB** = 10_B	P02_IOCR0.PC3 = $0XXXX_B$	In
RXDB3/ P14.1	ERAY_**CUST1.RISB** = 11_B	P14_IOCR0.PC1 = $0XXXX_B$	In
TXDB/ P02.2	not applicable	P02_IOCR0.PC2 = $1X110_B$	Out
TXDB/ P14.0	not applicable	P14_IOCR0.PC0 = $1X100_B$	Out
TXDB/ P14.5	not applicable	P14_IOCR4.PC5 = $1X110_B$	Out
TXDB/ P11.12	not applicable	P11_IOCR12.PC12 = $1X100_B$	Out
$\overline{\text{TXENB}}$/ P02.5	not applicable	P2_IOCR4.PC5 = $1X110_B$	Out
$\overline{\text{TXENB}}$/ P14.6	not applicable	P14_IOCR4.PC6 = $1X110_B$	Out
$\overline{\text{TXENB}}$/ P11.11	not applicable	P11_IOCR8.PC11 = $1X110_B$	Out
$\overline{\text{TXENB}}$/ P11.6	not applicable	P11_IOCR4.PC6 = $1X010_B$	Out

23.9.3 On-Chip Connections

This section describes all on-chip interconnections of the E-Ray modules except the connections to I/O ports (see).

23.9.3.1 E-Ray Connections with IR

The E-Ray module of the TC21x/TC22x/TC23x has several on-chip interconnections to the IR. **Table 23-28** shows these interconnections. These enable the IR to handle different service request of E-Ray module via the DMA or Interrupt Service Routine.

Table 23-28 Request Assignment for IR

Line #	ERAY Output Signal	IR Request Input Line
00	INT0	SRC_ERAYINT0
01	INT1	SRC_ERAYINT1
02	TINT0	SRC_ERAYTINT0
03	TINT1	SRC_ERAYTINT1
04	NDAT0	SRC_ERAYNDAT0
05	NDAT1	SRC_ERAYNDAT1
06	MBSC0	SRC_ERAYMBSC0
07	MBSC1	SRC_ERAYMBSC1
08	OBUSY	SRC_ERAYOBUSY
09	IBUSY	SRC_ERAYIBUSY

23.9.3.2 E-Ray Connections with SMU

The E-Ray module of the TC21x/TC22x/TC23x provides to the SMU the following alarms (ALMx): SRAM ECC single bit correction, SRAM ECC uncorrectable error, SRAM address error, SRAM buffer address error..

23.9.3.3 E-Ray Connections with the External Request Unit of SCU

The E-Ray module of the TC21x/TC22x/TC23x has several on-chip interconnections to the External Request Unit (ERU) in the SCU to externally trigger stop watch events and to provide a global time e.g. to the on chip timers. **Table 23-29** and **Table 23-30** show these interconnections.

Table 23-29 External Stop Watch Request Assignment

ERAY Input Signal	ERU Request Output Line	Selected by
STPWT0	ERU_PDOUT0	**CUST1.STPWTS** = 00_B
STPWT1	ERU_PDOUT1	**CUST1.STPWTS** = 01_B
STPWT2	ERU_PDOUT2	**CUST1.STPWTS** = 10_B
STPWT3	ERU_PDOUT3	**CUST1.STPWTS** = 11_B

Table 23-30 Global Macrotick Connection to ERU

ERAY Output Signal	ERU Request Input Line	Selected by
MT	ERU_IN23	ERU_EICR1.EXIS0 = 11_B

23.9.3.4 E-Ray Connections to GTM

The E-Ray module of the TC21x/TC22x/TC23x has several on-chip interconnections to the Generic Timer Module (GTM). **Table 23-31** show these interconnection s.

Table 23-31 Global Macrotick Connection to GTM

ERAY Output Signal	TIM Input Line
MT	TIM0_7

23.9.3.5 E-Ray Connections with the External Clock Output of SCU

The E-Ray module of the TC21x/TC22x/TC23x has one on-chip interconnections to the External Clock Output Unit in the SCU to distribute externally as also internally the Macro Tick as time base for distributed system control. **Table 23-32** shows this interconnection.

Table 23-32 Global Macrotick Connection to External Clock Output

ERAY Output Signal	External Clock Output	Selected by
MT0	f_{MT0}	SCU_EXTCON.SEL0 = 1111_B

23.9.4 OCDS Trigger Bus (OTGB) Interface

The E-Ray module has two 16 bit and one 32 bit Trigger Sets (**Table 23-33**) which are selected with the **OTSS** register.

Table 23-33 E-Ray Trigger Sets

Trigger Set	Details
TS16_SEP Service Requests, Errors and POC State	Table 23-35
TS16_MC Macrotick Counter	Table 23-36
TS32_SCSC State, Cycle and Slot Counter	Table 23-37

Table 23-34 shows all possible Trigger Set mapping options. If OTGB0 and/or OTGB1 is not used for E-Ray, Trigger Sets of other sources can be added in the OTGM module.

Table 23-34 Trigger Set Mapping Options

Width	OTGB0	OTGB1	OTGB2
32 Bit	TS16_SEP/MC		
		TS16_SEP/MC	
	TS16_SEP/MC	TS16_SEP/MC	
64 Bit			TS32_SCSC
	TS16_SEP/MC		TS32_SCSC
		TS16_SEP/MC	TS32_SCSC
	TS16_SEP/MC	TS16_SEP/MC	TS32_SCSC

Table 23-35 TS16_SEP Service Requests, Errors and POC State

Bits	Description
0	Interrupt 0 Service Request (INT0SRC)
1	Interrupt 1 Service Request (INT1SRC)
2	Timer Interrupt 0 Service Request (TINT0SRC)
3	Timer Interrupt 1 Service Request (TINT1SRC)
4	New Data 0 Service Request (NDAT0SRC)
5	New Data 1 Service Request (NDAT1SRC)
6	Message Buffer Status Changed 0 Service Request (MBSC0SRC)
7	Message Buffer Status Changed 1 Service Request (MBSC1SRC)

Table 23-35 TS16_SEP Service Requests, Errors and POC State (cont'd)

Bits	Description
8	Output Buffer Busy Service Request (OBUSYSRC)
9	Input Buffer Busy Service Request (IBUSYSRC)
10	Reserved
11	Error on Channel A (EIR.EDA)
12	Error on Channel B (EIR.EDB)
[15:13]	POC State bits[2:0] (CCSV.POCS)

Table 23-36 TS16_MC Macrotick Counter

Bits	Description
[13:0]	Macrotick Value (MTCCV.MTV)
[15:14]	Reserved

Table 23-37 TS32_SCSC State, Cycle and Slot Counter

Bits	Description
[10:0]	Slot Counter Channel A (SCV.SCCA)
[22:12]	Slot Counter Channel B (SCV.SCCB)
[29:24]	Cycle Counter Value (MTCCV.CCV)
30	Transfer Input Buffer Completed (SIR.TIBC)
31	Transfer Output Buffer Completed (SIR.TOBC)
11,23	Reserved

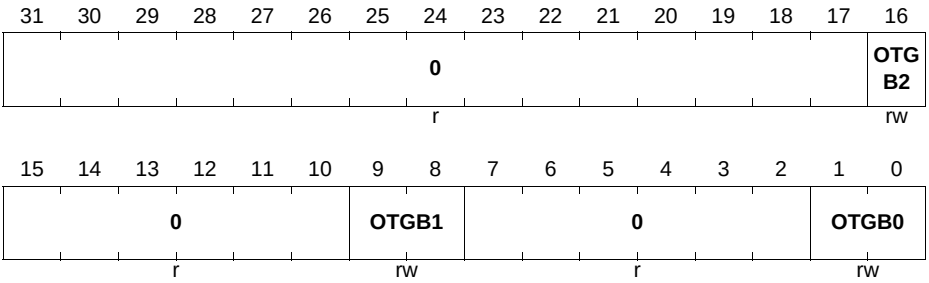
TS32_SCSC becomes valid (posedge on otgb2_valid_o) on a change of any Slot Counter or Transfer Buffer state. This covers also the Cycle Counter which will always change with a slower rate.

23.9.5 OTGB E-Ray Registers

23.9.5.1 OCDS Trigger Bus (OTGB)

The OTGB control register is cleared by Debug Reset. Write access is 32 bit wide only and requires Supervisor Mode.

OTSS
OCDS Trigger Set Select **(0870_H)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
OTGB0	[1:0]	rw	**Trigger Set for OTGB0** 0_D No Trigger Set selected 1_D Trigger Set TS16_SEP (**Table 23-35**) 2_D Trigger Set TS16_MC (**Table 23-36**) 3_D reserved
OTGB1	[9:8]	rw	**Trigger Set for OTGB1** 0_D No Trigger Set selected 1_D Trigger Set TS16_SEP (**Table 23-35**) 2_D Trigger Set TS16_MC (**Table 23-36**) 3_D reserved
OTGB2	16	rw	**Trigger Set for OTGB2** 0_D No Trigger Set selected 1_D Trigger Set TS32_SCSC (**Table 23-37**)
0	[7:2], [15:10], [31:17]	r	**Reserved** Read as 0; must be written with 0.

23.9.6 BPI_FPI Module Registers

Note: Register bits marked "r" in the following register description are virtual registers and do not contain flip-flops. They are always read as 0.

Clock Control Register (CLC)

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI_FPI for the module. Where a module kernel is connected to the CLC clock control interface, CLC controls the module clock, sleep mode and disable mode for the module.

CLC
Clock Control Register **(0000_H)** **Reset Value: 0000 0003_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					RMC			0				EDIS	0	DISS	DISR
r					rw			r				rw	r	rh	rw

Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. *Note: This bit disables the kernel clocks f_{CLC_ERAY} and the sampling clock f_{SCLK}.*
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module.
EDIS	3	rw	**External Sleep Mode Request Disable Bit** Used to control module's sleep mode. *Note: If this bit is cleared the kernel clock f_{CLC_ERAY} and the sampling clock f_{SCLK} are disabled during System Sleep Mode.*

Field	Bits	Type	Description
RMC	[10:8]	rw	**Clock Divider in Run Mode** 000_B No clock signal f_{CLC_ERAY} generated (default after reset) 001_B Clock $f_{CLC_ERAY} = f_{SPB}$ selected 010_B Clock $f_{CLC_ERAY} = f_{SPB}$ /2 selected 011_B Clock $f_{CLC_ERAY} = f_{SPB}$ /3 selected 100_B Clock $f_{CLC_ERAY} = f_{SPB}$ /4 selected 101_B Clock $f_{CLC_ERAY} = f_{SPB}$ /5 selected 110_B Clock $f_{CLC_ERAY} = f_{SPB}$ /6 selected 111_B Clock $f_{CLC_ERAY} = f_{SPB}$ /7 selected *Note: This bit field is not affected by an application reset.* *Note: This bit field only controls the kernel clock f_{CLC_ERAY} and not the sampling clock f_{SCLK}.*
0	[31:16], [15:11], [7:4], 2	r	**Reserved** Read as 0; should be written with 0.

Note: The number of module clock cycles (wait states) which are required by the kernel to execute a read or write access depends on the selected CLC clock frequency, which is selected via bit field RMC in the CLC register. Therefore, increasing CLC.RMC may result in a longer FPI Bus read cycle access time for kernel registers and can also slow down the write throughput to the kernel registers.

OCDS Control and Status Register (OCS)

The OCDS Control and Status (OCS) register is cleared by Debug Reset. The OCS register can only be written when the OCDS is enabled. If OCDS is being disabled, the OCS register value will not change. When OCDS is disabled the OCS suspend control is ineffective. Write access is 32 bit wide only and requires Supervisor Mode.

OCS
OCDS Control and Status **(08E8$_H$)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**SUS STA**	**SUS _P**	**SUS**				**0**							
r		rh	w	rw				r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
SUS	[27:24]	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0_H Will not suspend 1_H Hard suspend. Clocks f_{CLC_ERAY} and the sampling clock f_{SCLK} are switched off immediately. No read or write access to any registers. 2_H Soft suspend. This bit forces the module into freeze state. **others**, reserved
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	[23:0], [31:30]	r	**Reserved** Read as 0; must be written with 0.

Access Enable Register (ACCEN0)

The Access Enable Register 0 controls write access[1)] for transactions with the on chip bus master TAG ID 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000_B, EN1 -> TAG ID 000001_B, ... , EN31 -> TAG ID 011111_B.

ACCEN0
Access Enable Register 0 ($08FC_H$) Reset Value: FFFF FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN 31	**EN 30**	**EN 29**	**EN 28**	**EN 27**	**EN 26**	**EN 25**	**EN 24**	**EN 23**	**EN 22**	**EN 21**	**EN 20**	**EN 19**	**EN 18**	**EN 17**	**EN 16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN 15	**EN 14**	**EN 13**	**EN 12**	**EN 11**	**EN 10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n = 0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

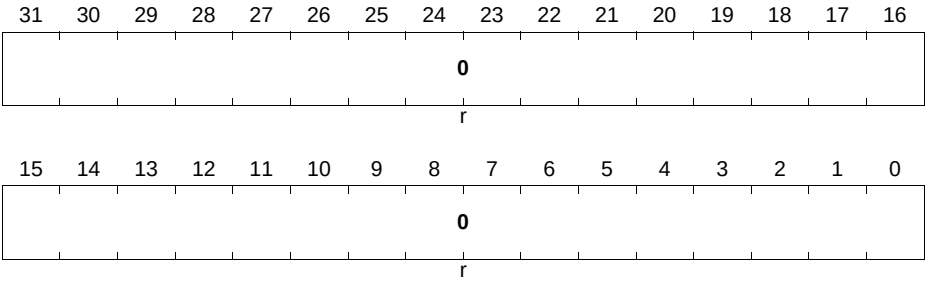
Access Enable Register (ACCEN1)

The Access Enable Register 1 controls write access[1)] for transactions with the on chip bus master TAG ID 100000_B to 111111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the AURIX devices.

1) The BPI_FPI Access Enable functionality controls only write transactions to the CLC, OCS, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000_B, EN1 -> TAG ID 100001_B, ... , EN31 -> TAG ID 111111_B.

ACCEN1
Access Enable Register 1 **($08F8_H$)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
0	[31:0]	r	**Reserved** Read as 0; should be written with 0.

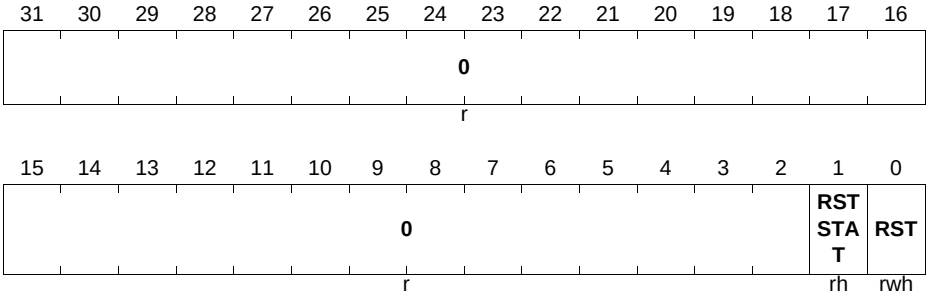
Kernel Reset Register 0 (KRST0)

The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers.The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the KRSTCLE.CLR register bit.

Note: During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

KRST0
Kernel Reset Register 0 **(08F4$_H$)** **Reset Value: 0000 0000$_H$**

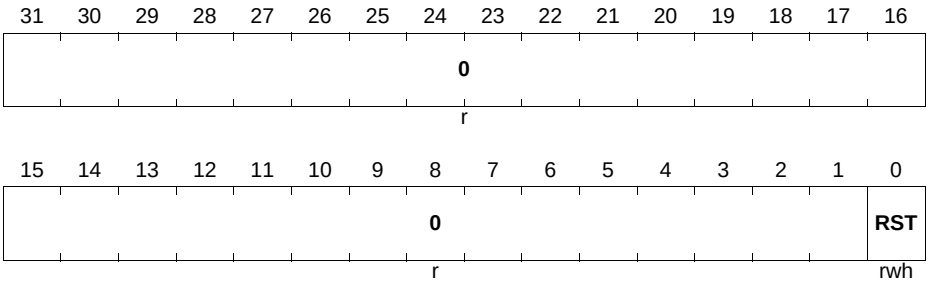


Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed.
RSTSTAT	1	rw	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. 0_B No kernel reset was executed 1_B Kernel reset was executed This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register.
0	[31:2]	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1 (KRST1)

The Kernel Reset Register 1 is used to reset the related module kernel. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers (KRSTx1.RST and KRSTx0.RST) related to the module kernel that should be reset (kernel 0 or kernel 1). The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

KRST1
Kernel Reset Register 1 **($08F0_H$)** **Reset Value: $0000\ 0000_H$**

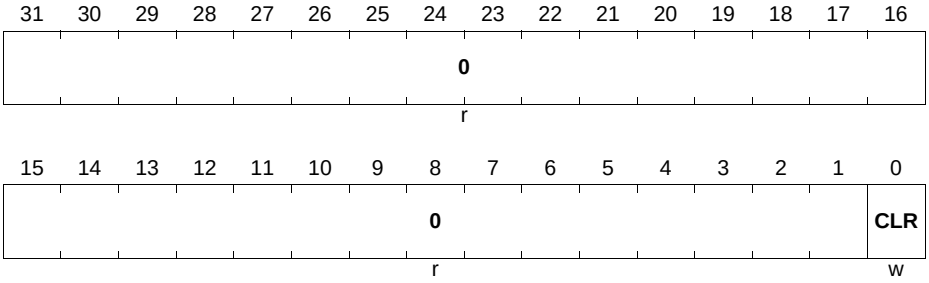


Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Status Clear Register (KRSTCLR)

The Kernel Reset Register Clear register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR
Kernel Reset Status Clear Register (08EC$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT Read always as 0.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

23.9.7 Interrupt Registers

Two different type of Interrupt Registers are described within this chapter.
The Interrupt Control register enable the selection of the Service Request used to signal an event. The Interrupt Control registers **NDIC1** to **NDIC4** select the service request node used for New Data Events.The Interrupt Control registers **MSIC1** to **MSIC4** select the service request node used for Message Buffer Status Changed Events.
The Interrupt Service Request Control Registers control the eight service request nodes.

New Data Interrupt Control 1 (NDIC1)

This New Data Interrupt Control register controls the interrupt that becomes active (NDAT0SRC or NDAT1SRC) on a ND flag turning active of all configured Message Buffers 0 to Message Buffers 31.

NDIC1
New Data Interrupt Control 1 ($03A8_H$) Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NDIP 31	**NDIP 30**	**NDIP 29**	**NDIP 28**	**NDIP 27**	**NDIP 26**	**NDIP 25**	**NDIP 24**	**NDIP 23**	**NDIP 22**	**NDIP 21**	**NDIP 20**	**NDIP 19**	**NDIP 18**	**NDIP 17**	**NDIP 16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDIP 15	**NDIP 14**	**NDIP 13**	**NDIP 12**	**NDIP 11**	**NDIP 10**	**NDIP 9**	**NDIP 8**	**NDIP 7**	**NDIP 6**	**NDIP 5**	**NDIP 4**	**NDIP 3**	**NDIP 2**	**NDIP 1**	**NDIP 0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
NDIPn (n = 0-31)	n	rw	**New Data Interrupt Pointer n (n = 0-31)** NDIPn determines the interrupt (NDAT0SRC or NDAT1SRC) of the service request output that becomes active on a New Data Flag becoming active. 0_B NDAT0SRC selected for New Data Service Request 1_B NDAT1SRC selected for New Data Service Request

New Data Interrupt Control 2 (NDIC2)

This New Data Interrupt Control register controls the interrupt that becomes active (NDAT0SRC or NDAT1SRC) on a ND flag turning active of all configured Message Buffers 32 to Message Buffers 63.

NDIC2
New Data Interrupt Control 2 **($03AC_H$)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NDIP 63	**NDIP 62**	**NDIP 61**	**NDIP 60**	**NDIP 59**	**NDIP 58**	**NDIP 57**	**NDIP 56**	**NDIP 55**	**NDIP 54**	**NDIP 53**	**NDIP 52**	**NDIP 51**	**NDIP 50**	**NDIP 49**	**NDIP 48**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDIP 47	**NDIP 46**	**NDIP 45**	**NDIP 44**	**NDIP 43**	**NDIP 42**	**NDIP 41**	**NDIP 40**	**NDIP 39**	**NDIP 38**	**NDIP 37**	**NDIP 36**	**NDIP 35**	**NDIP 34**	**NDIP 33**	**NDIP 32**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
NDIPn (n = 32-63)	n - 32	rw	**New Data Interrupt Pointer n (n = 32-63)** NDIPn determines the interrupt (NDAT0SRC or NDAT1SRC) of the service request output that becomes active on a New Data Flag becoming active. 0_B NDAT0SRC selected for New Data Service Request 1_B NDAT1SRC selected for New Data Service Request

New Data Interrupt Control 3 (NDIC3)

This New Data Interrupt Control register controls the interrupt that becomes active (NDAT0SRC or NDAT1SRC) on a ND flag turning active of all configured Message Buffers 64 to Message Buffers 95.

NDIC3
New Data Interrupt Control 3 ($03B0_H$) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NDIP 95	**NDIP 94**	**NDIP 93**	**NDIP 92**	**NDIP 91**	**NDIP 90**	**NDIP 89**	**NDIP 88**	**NDIP 87**	**NDIP 86**	**NDIP 85**	**NDIP 84**	**NDIP 83**	**NDIP 82**	**NDIP 81**	**NDIP 80**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDIP 79	**NDIP 78**	**NDIP 77**	**NDIP 76**	**NDIP 75**	**NDIP 74**	**NDIP 73**	**NDIP 72**	**NDIP 71**	**NDIP 70**	**NDIP 69**	**NDIP 68**	**NDIP 67**	**NDIP 66**	**NDIP 65**	**NDIP 64**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
NDIPn (n = 64-95)	n - 64	rw	**New Data Interrupt Pointer n (n = 64-95)** NDIPn determines the interrupt (NDAT0SRC or NDAT1SRC) of the service request output that becomes active on a New Data Flag becoming active. 0_B NDAT0SRC selected for New Data Service Request 1_B NDAT1SRC selected for New Data Service Request

New Data Interrupt Control 4 (NDIC4)

This New Data Interrupt Control register controls the interrupt that becomes active (NDAT0SRC or NDAT1SRC) on a ND flag turning active of all configured Message Buffers 96 to Message Buffers 127.

NDIC4
New Data Interrupt Control 4 ($03B4_H$) Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NDIP 127	**NDIP 126**	**NDIP 125**	**NDIP 124**	**NDIP 123**	**NDIP 122**	**NDIP 121**	**NDIP 120**	**NDIP 119**	**NDIP 118**	**NDIP 117**	**NDIP 116**	**NDIP 115**	**NDIP 114**	**NDIP 113**	**NDIP 112**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDIP 111	**NDIP 110**	**NDIP 109**	**NDIP 108**	**NDIP 107**	**NDIP 106**	**NDIP 105**	**NDIP 104**	**NDIP 103**	**NDIP 102**	**NDIP 101**	**NDIP 100**	**NDIP 99**	**NDIP 98**	**NDIP 97**	**NDIP 96**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
NDIPn (n = 96-127)	n - 96	rw	**New Data Interrupt Pointer n (n = 96-127)** NDIPn determines the interrupt (NDAT0SRC or NDAT1SRC) of the service request output that becomes active on a New Data Flag becoming active. 0_B NDAT0SRC selected for New Data Service Request 1_B NDAT1SRC selected for New Data Service Request

Message Buffer Status Changed Interrupt Control 1 (MSIC1)

This Message Buffer Status Change Interrupt Control register controls the interrupt that becomes active (MBSC0SRC or MBSC1SRC) on a MBC flag of all configured Message Buffer 0 to Message Buffer 31 turning active.

MSIC1
Message Buffer Status Changed Interrupt Control 1
($03B8_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSIP 31	**MSIP 30**	**MSIP 29**	**MSIP 28**	**MSIP 27**	**MSIP 26**	**MSIP 25**	**MSIP 24**	**MSIP 23**	**MSIP 22**	**MSIP 21**	**MSIP 20**	**MSIP 19**	**MSIP 18**	**MSIP 17**	**MSIP 16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSIP 15	**MSIP 14**	**MSIP 13**	**MSIP 12**	**MSIP 11**	**MSIP 10**	**MSIP 9**	**MSIP 8**	**MSIP 7**	**MSIP 6**	**MSIP 5**	**MSIP 4**	**MSIP 3**	**MSIP 2**	**MSIP 1**	**MSIP 0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
MSIPn (n = 0-31)	n	rw	**Message Buffer Status Changed Interrupt Pointer n (n = 0-31)** MSIPn determines the interrupt (MBSC0SRC or MBSC1SRC) of the service request output that becomes active on a Message Buffer Status Changed Flag becoming active. 0_B MBSC0SRC selected for Message Buffer Status Changed Service Request 1_B MBSC1SRC selected for Message Buffer Status Changed Service Request

Message Buffer Status Changed Interrupt Control 2 (MSIC2)

This Message Buffer Status Change Interrupt Control register controls the interrupt that becomes active (MBSC0SRC or MBSC1SRC) on a MBC flag of all configured Message Buffer 32 to Message Buffer 63 turning active.

MSIC2
Message Buffer Status Changed Interrupt Control 2
($03BC_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSIP 63	**MSIP 62**	**MSIP 61**	**MSIP 60**	**MSIP 59**	**MSIP 58**	**MSIP 57**	**MSIP 56**	**MSIP 55**	**MSIP 54**	**MSIP 53**	**MSIP 52**	**MSIP 51**	**MSIP 50**	**MSIP 49**	**MSIP 48**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSIP 47	**MSIP 46**	**MSIP 45**	**MSIP 44**	**MSIP 43**	**MSIP 42**	**MSIP 41**	**MSIP 40**	**MSIP 39**	**MSIP 38**	**MSIP 37**	**MSIP 36**	**MSIP 35**	**MSIP 34**	**MSIP 33**	**MSIP 32**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
MSIPn (n = 32-63)	n - 32	rh	**Message Buffer Status Changed Interrupt Pointer n (n = 32-63)** MSIPn determines the interrupt (MBSC0SRC or MBSC1SRC) of the service request output that becomes active on a Message Buffer Status Changed Flag becoming active. 0_B MBSC0SRC selected for Message Buffer Status Changed Service Request 1_B MBSC1SRC selected for Message Buffer Status Changed Service Request

Message Buffer Status Changed Interrupt Control 3 (MSIC3)

This Message Buffer Status Change Interrupt Control register controls the interrupt that becomes active (MBSC0SRC or MBSC1SRC) on a MBC flag of all configured Message Buffer 64 to Message Buffer 95 turning active.

MSIC3
Message Buffer Status Changed Interrupt Control 3
($03C0_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSIP 95	**MSIP 94**	**MSIP 93**	**MSIP 92**	**MSIP 91**	**MSIP 90**	**MSIP 89**	**MSIP 88**	**MSIP 87**	**MSIP 86**	**MSIP 85**	**MSIP 84**	**MSIP 83**	**MSIP 82**	**MSIP 81**	**MSIP 80**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSIP 79	**MSIP 78**	**MSIP 77**	**MSIP 76**	**MSIP 75**	**MSIP 74**	**MSIP 73**	**MSIP 72**	**MSIP 71**	**MSIP 70**	**MSIP 69**	**MSIP 68**	**MSIP 67**	**MSIP 66**	**MSIP 65**	**MSIP 64**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
MSIPn (n = 64-95)	n - 64	rw	**Message Buffer Status Changed Interrupt Pointer n (n = 64-95)** MSIPn determines the interrupt (MBSC0SRC or MBSC1SRC) of the service request output that becomes active on a Message Buffer Status Changed Flag becoming active. 0_B MBSC0SRC selected for Message Buffer Status Changed Service Request 1_B MBSC1SRC selected for Message Buffer Status Changed Service Request

Message Buffer Status Changed Interrupt Control 4 (MSIC4)

This Message Buffer Status Change Interrupt Control register controls the interrupt that becomes active (MBSC0SRC or MBSC1SRC) on a MBC flag of all configured Message Buffer 96 to Message Buffer 127 turning active.

MSIC4
Message Buffer Status Changed Interrupt Control 4
($03C4_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSIP 127	**MSIP 126**	**MSIP 125**	**MSIP 124**	**MSIP 123**	**MSIP 122**	**MSIP 121**	**MSIP 120**	**MSIP 119**	**MSIP 118**	**MSIP 117**	**MSIP 116**	**MSIP 115**	**MSIP 114**	**MSIP 113**	**MSIP 112**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSIP 111	**MSIP 110**	**MSIP 109**	**MSIP 108**	**MSIP 107**	**MSIP 106**	**MSIP 105**	**MSIP 104**	**MSIP 103**	**MSIP 102**	**MSIP 101**	**MSIP 100**	**MSIP 99**	**MSIP 98**	**MSIP 97**	**MSIP 96**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
MSIPn (n = 96-127)	n - 96	rw	**Message Buffer Status Changed Interrupt Pointer n (n = 96-127)** MSIPn determines the interrupt (MBSC0SRC or MBSC1SRC) of the service request output that becomes active on a Message Buffer Status Changed Flag becoming active. 0_B MBSC0SRC selected for Message Buffer Status Changed Service Request 1_B MBSC1SRC selected for Message Buffer Status Changed Service Request

23.10 Revision History

This User's Manual is based on the IP Specification: **Revision 1.2.6**.

Table 23-38 Revision History

Version Number	Changes to Previous Version
Rev_0.1	First Draft
Rev_0.2	Adapted to PWD 0.8
Rev_0.3	Chapters 3, 5, 6 completed
Rev_0.4	Adapted to actual state of protocol development
Rev_0.5	Adapted to actual state of protocol development
Rev_0.51	Adapted to actual state of protocol development
Rev_0.52	Adapted to actual state of protocol development
Rev_0.53	Adapted to actual state of protocol development
Rev_0.6	Adapted to actual state of protocol development
Rev_0.61	Adapted to actual state of protocol development
Rev_0.62	Adapted to actual state of protocol development
Rev_1.0	First complete revision
Rev_1.01	Message Buffer Status bits PLE, MLST, ES replaced by bits ESA, ESB, MLST
Rev_1.02	IBCR, IBCM, OBCR, OBCM: addresses changed HDC2: register removed MHDC1: renamed to MHDC Message Buffer 0 dedicated to hold key slot ID SFS: description updated ESIDn, OSIDn: description updated EIR: bit SCE removed EILS: bit SCEL removed EIES, EIER: bit SCEE removed

Table 23-38 Revision History (cont'd)

Version Number	Changes to Previous Version
Rev_1.2	IDEFAULT_CONFIG added to POC working CCSV: assignment of states to $_{\text{POCS changed:}}$ $_{\text{POCS}}$ = 00 0000_B = "DEFAULT_CONFIG" POCS = 00 1111_B = "CONFIG" CCSV: bit DCREQ removed SIR: bit MBSI added SILS: bit MBSIL added SIES, SIER: bit MBSIE added Register BGSC removed EIR: bits SMEB, SMEA removed EILS: bits SMEBL, SMEAL removed EIES, EIER: bits SMEBE, SMEAE removed Registers TXRQ3, TXRQ4, NDAT3, NDAT4,MBSC3, MBSC4 added Bus guardian related pins eray_arm, eray_bgt, eray_mt, eray_bge1, and eray_bge2 have no function PRTC1: Configuration parameter CASM added WRHS1: Bit NME changed to PPIT RDHS1: Bit NME changed to PPIT Pin eray_scanmode for scan mode control added
Rev_1.3	Changed "r" bits into "rh" bits. Made ACS Register writable. Included Reserved Bits into EIES and EIER Changed access write from "rw" to "rwh" for Register CUST1.IEN. Changed access type of unused bits (0) to "rw". Changed access type of CUST1.INT0 to rwh. Changed Reset value of CUST0 to C104 0105_H Updated the "Known Limitations of E-Ray IP-Module", revision 4.07.2005.

Table 23-38 Revision History (cont'd)

Version Number	Changes to Previous Version
Rev_1.4	Updated the "Errata of E-Ray IP-Module", revision 1.0 30.06.2006. Renamed from "Interrupt Flag Interface" to "Internal Signal and Flag Interface" With this revision it is possible to use Message Buffer 1for SYNC Frame transmission in addition to Message Buffer 0 if SYNC Frames should have different payloads on channel A and B TEST1: Bit ELBE for control of internal / external loop back mode added, description of internal loop back added EIR: Handling of bits PERR and RFO same as for other bits, bit MHF added SIR: Bit RFF renamed to RFCL, handling of bits RFNE, RFCL same as for other bits EILS: Bit MHFL added, SILS: Bit RFFL renamed to RFCLL EIES, EIER: Bit MHFE added SIES, SIER: Bit RFFE renamed to RFCLE Register STPW renamed to STPW1 STPW2: Register added CCSV: Bits PSL added SWNIT: Bits MTSA, MTSB added MRC: Bit SPLM added FCL: Register added FSR: Register added MHDF: Register added MBSC1/2/3/4: Naming of bits changed from MBS to MBC to distinguish between Message Buffer status flag (MBC) and Message Buffer status register (MBS) CREL: Register added ENDN: Register added Message Buffers in Message RAM: Header 2 and 3 updated from received Data Frames only MBS: Bits FTA, FTB, CCS, RCIS, SFIS, SYNS,NFIS, PPIS, RESS added Description Asynchronous Transmit Mode, added: …This write operation has to be directly preceded by two consecutive write accesses to the Configuration LockKey (unlock sequence)… Description Loop Back Mode, added: …This write operation has to be directly preceded by two consecutive write accesses to the Configuration LockKey (unlock sequence).

Table 23-38 Revision History (cont'd)

Version Number	Changes to Previous Version
Rev_1.4 (cont'd)	Description LCK.CLK, corrected: To leave CONFIG state by writing SUCC1.CMD (commands READY, MONITOR_MODE, ATM,LOOP_BACK), …Third write: SUCC1.CMD Description EIR.MHF, added: The flag signals a Message Handler constraints violation condition. It is set whenever one of the flags MHDF.SNUA, MHDF.SNUB, MHDF.FNFA, MHDF.FNFB,MHDF.TBFA, MHDF.TBFB, MHDF.WAHP changes from 0 to 1. Description SIR.CAS, added: This flag is set by the Communication Controller during STARTUP state when a CAS or a potential CAS was received. 1 = Bit pattern matching the CAS symbol received 0 = No bit pattern matching the CAS symbol received. Description SIR.MBSI, corrected: This flag is set by the Communication Controller when the Message Buffer status MBS has changed if bit MBI of that Message Buffer is set. Description T0C, added: Note: The configuration of timer 0 is compared against the Macrotick counter value, there is no separate counter for timer 0… Description SUCC1.CMD, modified: Note included into description of CHI command CLEAR_RAMS. …Access to the configuration and status registers is possible during execution of CHI command CLEAR_RAMS… Description SUCC1.TSM, changed: …The key slot ID is configured in the Header Section of Message Buffer 0 respectively Message Buffers 0 and 1 depending on bit MRC.SPLM. In case TSM = 1, Message Buffer 0 respectively Message Buffers 0,1 can be (re)configured in "DEFAULT_CONFIG" or "CONFIG" state only. In ALL slot mode the Communication Controller may transmit in all slots. TSM is a configuration bit which can only be set / reset by the Host. The bit can be written in "DEFAULT_CONFIG" or "CONFIG" state only. The Communication COntroller changes to ALL slot mode when the Host successfully applied the ALL_SLOTS command by writing CMD = "0101" in POC states "NORMAL_ACTIVE" or "NORMAL_PASSIVE"... Description PRTC1.SPP, added: Note: The current revision 2.1 of the FlexRay™ protocol requires that SPP = "00". The alternate strobe point positions could be used to compensate for asymmetries in the physical layer. Description CCSV.CSNI, corrected: …Reset by CHI command RESET_STATUS_INDICATORS or by transition from "HALT" to "DEFAULT_CONFIG" state or from "READY" to "STARTUP" state.

Table 23-38 Revision History (cont'd)

Version Number	Changes to Previous Version
Rev_1.4 (cont'd)	Description CCSV.CSAI, corrected: …Reset by CHI command RESET_STATUS_INDICATORS or by transition from "HALT" to "DEFAULT_CONFIG" state or from "READY" to "STARTUP" state. Description CCSV.WSV, corrected: …Reset by CHI command RESET_STATUS_INDICATORS or by transition from "HALT" to "DEFAULT_CONFIG" state or from "READY" to "STARTUP" state. Description SFS.VSAE, modified: Holds the number of valid SYNC Frames received on channel A in the even communication cycle. If transmission of SYNC Frames is enabled by SUCC1.TXSY the value is incremented by one… Description SFS.VSAO, modified: Holds the number of valid SYNC Frames received on channel A in the odd communication cycle. If transmission of SYNC Frames is enabled by SUCC1.TXSY the value is incremented by one… Description SFS.VSBE, modified: Holds the number of valid SYNC Frames received on channel B in the even communication cycle. If transmission of SYNC Frames is enabled by SUCC1.TXSY the value is incremented by one… Description SFS.VSBO, modified: Holds the number of valid SYNC Frames received on channel B in the odd communication cycle. If transmission of SYNC Frames is enabled by SUCC1.TXSY the value is incremented by one… Description ACS, added: Note: The set condition of flags CIA and CIB is also fulfilled if there is only one single Frame in the slot and the slot boundary at the end of the slot is reached during the Frames channel idle recognition phase… Description MRC.SEC1:0, changed: …Exception: In nodes configured for SYNC Frame transmission or for single slot mode operation Message Buffer 0 (and if SPLM = 1, also Message Buffer 1) is always locked Description MRC.SPLM, changed: This bit is only evaluated if the node is configured as sync node (SUCC1.TXSY = 1) or for single slot mode operation (SUCC1.TSM = 1)… Description MRC.SPLM, changed: Note: In case the node is configured as sync node (SUCC1.TXSY = 1) or for single slot mode operation (SUCC1.TSM = 1), Message Buffer 0 resp. 1 is reserved for SYNC Frames or single slot Frames and have to be configured with the node-specific key slot ID. In case the node is neither configured as sync node nor for single slot operation Message Buffer 0 resp. 1 is treated like all other Message Buffers.

Table 23-38 Revision History (cont'd)

Version Number	Changes to Previous Version
Rev_1.4 (cont'd)	Description Parity Check, changed: …When a parity error occurs when the Message Handler reads a Frame with Network Management information (PPI = 1) from the Transient Buffer RAM A, B the corresponding Network Management vector register NMV1…3 is not updated from that Frame. Included TC1797 O Ports the Physical Layer Interface is connected to. Description MBS.MLST, added: The flag is set in case the Host did not read the message before the Message Buffer was updated from a received Data Frame… …The flag is reset by a Host write to the Message Buffer via IBF or when a new message is stored into the Message Buffer after the Message Buffers ND flag was reset by reading out the Message Buffer via OBF. Description MBS, corrected: Note: The FlexRay™ protocol specification requires that FTA, and FTB can only be reset by the Host… Description NULL Frame Transmission, changed: …In this case, no Message Buffer status MBS is updated.
Rev_1.5	Included BPI Register (Service Request Nodes: INT0SRC, INT1SRC, TINT0SRC, TINT1SRC, NDAT0SRC, NDAT0SRC, MBSC0SRC, MBSC1SRC, CLC). Included for New Data and Message Buffer Status Changed Flags eight Interrupt Select Register (NDIC0, NDIC1, NDIC2, NDIC3, MSIC0, MSIC1, MSIC2, MSIC3) Included new address range for TC1797. Included into CUST1 Register Control (IBF1PAG, IBF1PAG) and Status Bits (IBS) for multiple buffer control. Included for TC1797 the IO Ports the Physical Layer Interface is connected to. Included into CUST1 1 out of 4 multiplexer Select Input Control for RXA (CUST1.RISA), RXB (CUST1.RISB) and STPWT (CUST1.STPWTS). Included DMA Trigger connections for TC1797.
Rev_1.6	Included a description of the Delayed Write Scheme to OBF and IBF. Corrected an erroneous description of IBF1PAG and IBF2PAG concerning the write access for specific values of IBFS. Included a verification scheme to verify correct polarity of IBFS and logic for IBF1PAG and IBF2PAG.
Rev_1.7	Changed P6.7 to P6.10 to use only p6.PDR1 only. Included table summarizing access wait states.
Rev_1.8	Included a description of registering the reset when E-Ray module is disabled and executing the reset when it is enabled hereafter.

Table 23-38 Revision History (cont'd)

Version Number	Changes to Previous Version
Rev_1.9	Included IP Specification changes of Revision 1.2.3. **DescriptionTEST1, added**: When the E-Ray IP is operated in one of its test modes that requires WRTEN to be set (RAM Test Mode, I/O Test Mode, Asynchronous Transmit Mode, and Loop Back Mode) only the selected test mode functionality is available. The test functions are not available in addition to the normal operational mode functions, they change the functions of parts of the E-Ray module. Therefore normal operation as specified outside this chapter and as required by the FlexRay™ protocol specification and the FlexRay™ conformance test is not possible. Test mode functions may not be combined with each other or with FlexRay™ protocol functions. The test mode features are intended for hardware testing or for FlexRay™ bus analyzer tools. They are not intended to be used in FlexRay™ applications. **Description TEST1.ELBE, modified:** …Bit ELBE is evaluated only when POC is in loop back mode and test multiplexer control is in non-multiplexing mode TMC = 00. **Description TEST1.TMC, modified:** TMC Test Multiplexer Control 00, 11= Normal signal path (default); 01 = RAM Test Mode - Internal busses are multiplexed to make all RAM blocks of the E-Ray module directly accessible by the Host… **Description Loop Back Mode, corrected:** …Reading CCSV.POCS will return "00 1101" while the E-Ray module is in loop back mode… **Description Loop Back mode, completed:** …When the CC is in loop back mode, a loop back test is started by the Host writing a message to the Input Buffer and requesting the transmission by writing to register IBCR… **Description LCK.CLK, corrected:** …If the write sequence below is interrupted by other write accesses between the second write to the Configuration Lock Key and the write access to the SUCC1 register, the Communication Controller remains in CONFIG state and the sequence has to be repeated. **Description LCK.TMK, corrected:** …If the write sequence below is interrupted by other write accesses between the second write to the Test Mode Key and the write access to the TEST1 register, TEST1.WRTEN is not set to '1' and the sequence has to be repeated. **Description EIR.CCL, corrected:** The flag signals that the write access to the CHI command vector SUCC1.CMD was not successful because the execution of the previous CHI command has not yet completed… **Description SIR.MTSA,B, modified:** Media Access Test symbol received on channel A,B during the preceding symbol window…

Table 23-38 Revision History (cont'd)

Version Number	Changes to Previous Version
Rev_1.9 (cont'd)	**Description T1C, removed:** … In case the configured Macrotick count is not within the valid range, timer 1 will not start… **Description SUCC1.CMD, corrected:** …In case the previous CHI command has not yet completed, EIR.CCL is set to 1 together with EIR.CNA; the CHI command needs to be repeated. Except for HALT state, a POC state change command applied while the CC is already in the requested POC state will be ignored. **Description SUCC1.CMD, added:** ALLOW_COLDSTART; …When called in states DEFAULT_CONFIG, CONFIG, HALT, or MONITOR_MODE, CMD will be reset to 0000 = command_not_accepted… D**escription SUCC1.CMD, completed:** RESET_STATUS_ INDICATORS; …Flags internally evaluated in the actual POC state are not reset… **Description SUCC1.CMD, corrected:** MONITOR_MODE; …In this mode the CC is able to receive FlexRay™ Frames and wakeup pattern… **Description SUCC1.CMD, added:** Table added which references the CHI commands from the FlexRay™ Protocol Specification v2.1 (section 2.2.1.1, Table 2-2) to the E-Ray CHI command vector CMD. **Description CCSV.RCA, completed:** …The READY command resets this counter to the maximum number of coldstart attempts as configured by SUCC1.CSA. **Description SWNIT.MTSA,B, modified:** Media Access Test symbol received on channel A,B during the preceding symbol window… **Description ESID[1…15].RXEA,B completed:** …sorted in ascending order, … …If the node transmits a SYNC Frame in an even communication cycle by itself, register ESID1 holds the respective SYNC Frame ID as configured in Message Buffer 0 and flags RXEA, RXEB are set… RXEA,B Received / Configured Even Sync ID on Channel A,B Signals that a SYNC Frame corresponding to the stored even sync ID was received on channel A or that the node is configured to be a sync node with key slot = EID (ESID1 only). 1 = SYNC Frame received on channel A / node configured to transmit SYNC Frames; 0 = No SYNC Frame received on channel A / node not configured to transmit SYNC Frames.

Table 23-38 Revision History (cont'd)

Version Number	Changes to Previous Version
Rev_1.9 (cont'd)	**Description OSID[1…15].RXOA,B completed:** … sorted in ascending order, … …If the node transmits a SYNC Frame in an odd communication cycle by itself, register OSID1 holds the respective SYNC Frame ID as configured in Message Buffer 0 and flags RXOA, RXOB are set…RXOA,B Received / Configured Odd Sync ID on Channel A,B Signals that a SYNC Frame corresponding to the stored odd sync ID was received on channel A or that the node is configured to be a sync node with key slot = OID(OSID1 only). 1 = SYNC Frame received on channel A / node configured to transmit SYNC Frames;0 = No SYNC Frame received on channel A / node not configured to transmit SYNC Frames **Description FRF.FID, modified:** Determines the Frame ID to be rejected by the FIFO. With the additional configuration of register FRFM, the corresponding Frame ID filter bits are ignored, which results in further rejected Frame IDs. When FRFM.MFID is zero, a Frame ID filter value of zero means that no Frame ID is rejected. **Description MHDF.WAHP, added:** Outside DEFAULT_CONFIG and CONFIG state this flag is set by the CC when the message handler tries to write message data into the Header Partition of the Message RAM due to faulty configuration of a Message Buffer… **Description MONITOR_MODE, completed:** …In this mode the CC is able to receive FlexRay™ Frames and to detect wakeup pattern… **Modified the description of CUST1.POBFEN to:** Parity Error Reporting of Output Buffer (OBF1/OBF1) RAMs Enable/Test Disable; 0_B: Parity error in Output Buffer (OBF1,OBF2) RAMs is not reported to SCU nor to OBF1, OBF2 RAM wrapper.1_B: The parity error reporting and thus the activation of error signals to SCU as also to OBF1 and OBF2 RAM wrapper is enabled. **Modified the description of CUST1.PMBFEN to:** Parity Error Reporting of Message Buffer (MBF) RAMs Enable/Test Disable; 0_B: Parity error in Message Buffer (MBF) RAMs is not reported to SCU nor to MBF RAM wrapper.1_B: The parity error reporting and thus the activation of error signals to SCU as also to MBF RAM wrapper is enabled. **Modified the description of CUST1.PITBFEN to:** Parity Error Reporting of Message Buffer (IBF1, IBF2, TBF1, TBF2) RAMs Enable/Test Disable; 0_B: Parity error in Input Buffer and Transient Buffer (IBF1, IBF2, TBF1, TBF2) RAMs is not reported to SCU nor to IBF1, IBF2, TBF1, and TBF2 RAM wrapper.1_B: The parity error reporting and thus the activation of error signals to SCU as also to IBF1, IBF2, TBF1, and TBF2 RAM wrapper is enabled

Table 23-38 Revision History (cont'd)

Version Number	Changes to Previous Version
Rev_2.0	Modified the address of INT0SRC, INT1STC, TINT0SRC, TINT1SRC, NDIC1…NDIC4, MSIC1…MSIC4, NDAT_NDAT0SRC, NDAT1SRC, MBSC0SRC, MBSC1SRC. Cleared up the Index entries. Changed Address Range from to 00000000_H …$00007FFF_H$ to $F0010000_H$ …$F0017FFF_H$
Rev_2.1	Inserted additional information on the minimal eray_bclk required to work properly. Corrected some Typos in the port implementation. Renamed the Interrupt Control Registers due to some problems extracting register names by automatic tools. Changed in Figure 2-1 the names of the clock signals. Changed the Long Name of Register ACS to Aggregated Channel Status Register. Included Message Handler Timing Details. Modified the Clock Signal Names in Figure 2-37.
Rev_2.2	Modified the Pinning of TC1797: TXDA: P0.14, RXDA:P0.9; TXENA: P0.10; TXDB: P0.12; RXDB: P0.12; TXENB: P0.11. Changed the name of the Bit RFCLL in SILS Register.
Rev_2.3	Included IP Specification changes of Revision 1.2.3. Description write access to Data Section of IBF, changed: …If not all bytes of a 32-bit word have been written by the Host (8/16-bit access only), WRDSn holds partly undefined data. Description SIR.WST, completed: This flag is set when the wakeup status vector CCSV.WSV is changed by a protocol event. Description SIR.SWE, modified: This flag is set after a stop watch activation when the actual cycle counter and Macrotick value are stored in the Stop Watch register (see section Stop Watch Register 1 (STPW1)). Description STPW1.ESWT, corrected: …In single-shot mode this bit is reset to 0 after the actual cycle counter and Macrotick value are stored in the Stop Watch register. Description SUCC1.CMD3:0, changed: RESET_STATUS_INDICATORS Resets status flags CCSV.CSNI, CCSV.CSAI, and CCSV.WSV to their default values. May be called in POC state "READY". When called in any other state, CMD will be reset to 0000_B = command_not_accepted. Description CCSV.POC, added: …101011_B = STARTUP_SUCCESS state 101100_B … 111111_B = reserved

Table 23-38 Revision History (cont'd)

Version Number	Changes to Previous Version
Rev_2.3	Description SUCC1.MTSA,B, added: Note: MTSA,B may also be changed outside "DEFAULT_CONFIG" or "CONFIG" state when the write to SUCC1 register is directly preceded by the unlock sequence as described in chapter Lock Register (LCK). This may be combined with CHI command SEND_MTS… Description Communication Controller Status Registers, added: …The status vector may change faster than the Host can poll the status vector, depending on eray_bclk frequency. Description Communication Controller Status Registers, removed: …All internal counters and the Communication Controller status flags are reset when the Communication Controller transits from "CONFIG" to "READY" state. Description CCSV.FSI, changed: …Reset by transition from HALT to "DEFAULT_CONFIG" state. Description CCSV.HRQ, changed: …Reset by transition from HALT to DEFAULT_CONFIG state or when entering READY state. Description CCSV.SLM, changed: …Set to the value defined by SUCC1. TSM when entering "READY", "CONFIG", or "DEFAULT_CONFIG" state. Description CCSV.CSI, completed: …The flag is set whenever the POC enters READY state due to CHI command READY. Description CCSV.WSV, modified: …000_B = UNDEFINED. Wakeup not yet executed by the Communication Controller… Description CCSV.RCA, corrected: …The "RUN" command resets this counter to the maximum number of coldstart attempts as configured by SUCC1.CSA. Description Communication Controller Status Registers, removed: …Note: CHI command "RESET_STATUS_INDICATORS" (SUCC1.CMD = 1010_B) resets flags FSI, HRQ, CSNI, CSAI, the slot mode SLM, and the wakeup status WSV. Description CC Error Vector, changed: Reset by transition from "HALT" to "DEFAULT_CONFIG" state or when entering "READY" state... Description MONITOR_MODE, added: …In "MONITOR_MODE" the Communication Controller is not able to distinguish between CAS and MTS symbols. In case one of these symbols is received on one or both of the two channels, the flags SIR.MTSA resp. SIR.MTSB are set. SIR.CAS has no function in "MONITOR_MODE". Inserted remark to negate eray_ibusy and eray_obusy signal to simplify the usage for DMA requesting and interrupt triggering. Corrected Typo in CMD field of SUCC1.

Table 23-38 Revision History (cont'd)

Version Number	Changes to Previous Version
Rev_2.4	Updated ERAY Kernel Errata List (Bosch Errata List from November 24th, 2006: "Errata_E-Ray_R1.0_20061124")
Rev_2.5	**Modified RISA: Receive Input Select Channel A** 00_B Channel A receiver input RXDA0 selected 01_B Channel A receiver input RXDA1 selected 10_B Channel A receiver input RXDA2 selected 11_B Channel A receiver input RXDA3 selected **Modified RISB: Receive Input Select Channel B** 00_B Channel B receiver input RXDB0 selected 01_B Channel B receiver input RXDB1 selected 10_B Channel B receiver input RXDB2 selected 11_B Channel B receiver input RXDB3 selected Description MONITOR_MODE, removed: …In "MONITOR_MODE" the receive FIFO is not available. Description Message RAM Configuration Register, added: … Note: … In case two or more Message Buffers are assigned to slot 1 by use of cycle filtering, all of them must be located either in the "Static Buffers" or at the beginning of the "Static + Dynamic Buffers" section… Description CCSV.POC[5:0], added: …101011_B = "STARTUP_SUCCESS" state 101100_B…111111_B = reserved. Inserted text condition to commonly cover IP revision 1.0.0 and IP revision 1.0.1. Updated ERAY Kernel Errata List (Bosch Errata List from December 20th, 2006: "Errata_E-Ray_R1.0.0_20061220"). Included IP Specification changes of Revision 1.2.5. Description EIR.SFO, changed: Set when either the number of SYNC Frames received during the last communication cycle or the total number of SYNC Frames received during the last double cycle exceeds the maximum number of SYNC Frames as defined by GTUC2.SNM. Description SUCC1.CMD[3:0], changed: RESET_STATUS_INDICATORS … May be called in POC states "READY" and "STARTUP"…

Table 23-38 Revision History (cont'd)

Version Number	Changes to Previous Version
Rev_2.6	Description CCSV.SLM, changed: Indicates the actual slot mode of the POC in states "READY", "STARTUP", "NORMAL_ACTIVE", and "NORMAL_PASSIV"E. Default is SINGLE. Changes to ALL, depending on SUCC1.TSM. In "NORMAL_ACTIVE" or "NORMAL_PASSIVE" state the CHI command ALL_SLOTS will change the slot mode from SINGLE over ALL_PENDING to ALL. Set to SINGLE in all other states. Description SFS.VSAE, removed: … (vSyncFramesEvenA) Description SFS.VSAO, removed: … (vSyncFramesOddA) Description SFS.VSBE[3:0], removed:… (vSyncFramesEvenB) Description SFS.VSBO[3:0], removed: …(vSyncFramesOddB) Description MBS.RCIS, MBS.SFIS, MBS.SYNS, MBS.NFIS, MBS.PPIS, MBS.RESS, completed: For receive buffers (CFG = 0) the following status bits are updated from both valid data and NULL Frames. If no valid Frame was received, the previous value is maintained. For transmit buffers the flags have no meaning and should be ignored. Description Network Management, added: Note: …When the Communication Controller transits to "HALT" state, the cycle count is not incremented and therefore the NM Vector is not updated. In this case NMV1…3 holds the value from the cycle before. Description Configuration of the FIFO, added: Note: It is recommended to program the MBI bits of the Message Buffers belonging to the FIFO to 0 via WRHS1.MBI to avoid generation of RX interrupts… Updated the CLC reset value to 0000 0100_H. Included the ERAY trigger signals to the DMA channels 10 -17. Modified the stop watch clock input. Shifted them from GOUT to PDOUT. Removed a description of the non implemented fractional divider. Updated ERAY Kernel Errata List (Bosch Errata List from February 19th, 2007: "E-Ray_Errata_Sheet_20070219"). Connected Macrotick signal to ERU input IN23. Modified the DMA connection. Included IBUSY and OBUSY and TINT0. Included Service Request node for IBUSY and OBUSY. Modified the addresses of NDIC1, NDIC2, NDIC3, NDIC4, MSIC1, MSIC2, MSIC3, and MSIC4 to have a continues address range for the new service request registers OBUSYSRC and IBUSYSRC.
Rev_2.7	Removed Parity Bit generation and checking Added ECC generation and checking Wording and typos correction Signal name clean up (e.g. clock signals, ERAY internal signals etc.) CLC is endinit protected

Table 23-38 Revision History (cont'd)

Version Number	Changes to Previous Version
Rev_3.0	Included IP Specification changes of Revision 1.2.6 Inserted text condition to commonly cover IP revision 1.0.0, IP revision 1.0.1 and IP revision 1.0.2. It is not possible to change the Stop Watch Mode (STPW1.SWMS) during enabled stop watch trigger (STPW1.ESWT) Implemented AI00042258. It is not possible to distinguish between static and dynamic frames, because limited functions in Monitor Mode (FRF.RSS will be ignored, filtering not functional). Added note in TEST1L/H register about the change of TEST1H.CERA/B if accessing TEST1H or L Reset value for KSCFG changed to $0x0001_h$. Was the initial value $0x0000_h$. ECC control and status registers added (SEC_CON, SED_CON, DED_CON, ECCR, ECCW) Removed ECC control form CUST1 Removed ECC test description from Register TEST2 Address space enlarged from 2k to 4k Figure E-Ray Register Map contains the new registers
Rev_3.1	ECC control and status registers renamed (SEC_CON -> SECCON, SED_CON -> SEDCON, DED_CON -> DEDCON Corrected RDHS3H.0 length field Renamed STPW1, 2, 3, 4 to STPWT0, 1, 2, 3 16-Bit: Detailed KSCFG functionality MT connected to ERU_3B2 (was ERU_3A2 in V3.0 16-Bit: KSCFG moved to 0x0000h 16-Bit: MT connected to ERU_3B2 (was ERU_3A2 in V3.0) Completed Register Table with page numbers Corrected address spaces that are reserved for customer specific purposes
Rev_3.2	16-bit: "Implement DEDCON registers" "Description of register SECCONL, SEDCON, DEDCON": changed TBFA/B to TBF1/2 implemented individual ECC control over each RAM + implement KSCFG register
Rev_3.3	ECC: "Define Signal WRECC in E-Ray Register TEST1" with modification: TEST2 used like in original implementation for parity 32-Bit: ECC: Reset Values of SECCON, SEDCON, DEDCON = 0x007Fh = enabled

Table 23-38 Revision History (cont'd)

Version Number	Changes to Previous Version
Rev_3.4	16-bit: "Specify and implement Module ID Register for FlexRay like TC1797" 16-bit: SECCON, SEDCON, DEDCON = 0x0000h = disabled 16-bit: Layout and pagination changes for User Manual (no Change Bars) wording corrected for User Manual (with Change Bars) add High Registers to Register Table correct all markers for Index / Appendix corrected listing of all reserved addresses
Rev_3.5	16-bit: Hint, to reset IBUSY and OBUSY before enabling these interrupts Added detailed figure of implementation Added overview figure of module
Rev_3.6	TXDA, TXDB and TXEN no longer enumerated (docu fix) Typos fixed Remove Examples
Rev_3.7	Markers for key-word -index reduced to 3 levels
Rev_3.8	port mapping added 32-bit: CLC description corrected: "requesting the HW to ~~clear~~ set bit CLC.DISS" 32-bit: added hint about initial value of CLC 16-bit: ERU_3B2 used for FR_MT and not ERU3A2 corrected IOCR settings in Table of Port connections.
Rev_3.9	32-bit: added BPI Protection and Module Reset 32-bit: removed SRCs
Rev_3.10	32-bit: added trace interface 32-bit: added tagging
Rev_3.11	32-bit: updated BPI Registers Chapter
Rev_3.12	32-bit: changed MHDS.CRAM reset value to '0' added: Reset Value CUST1.IBFS changes after 2 clock cycles from '0' to '1'. added: Reset Value SUCC1.PBUSY changed from '1' to '0' changed base address to F001C000
Rev_3.13	32-bit: added port mapping added IR wiring, added SMU interconnection
Rev_3.14	32-bit: added 2nd Kernel (ERAY0 and ERAY1).
Rev_3.15	32-bit: added: MT output connection to GTM

Table 23-38 Revision History (cont'd)

Version Number	Changes to Previous Version
Rev_3.16	32-bit: removed OCS.TGS, TGB, TB_P. Changed OCS description to current implementation. Removed OTSS and OTGB description.
Rev_3.17	32-bit: updated pin out description (typo fix)

24 Generic Timer Module (GTM)

24.1 Introduction

This document is based on the GTM_IP Specification Revision 2.02.1 (Released on 30.1.2013) of the Robert Bosch GmbH.

24.1.1 Overview

This document is the specification for the Generic Timer Module (GTM). It contains a module framework with submodules of different functionality. These submodules can be combined in a configurable manner to form a complex timer module that serves different application domains and different classes within one application domain. Because of this scalability and configurability the timer is called generic.

The hardware submodules have dedicated functionalities, e.g. there are timer input modules where incoming signals can be captured and characterized together with a notion of time.

Each GTM is build up therefore with submodules coming from those four groups. The application class is defined by the amount of components of those submodules integrated into the implemented GTM.

24.1.2 Document Structure

The structure of this document is motivated out of the aforementioned submodule classes. **Section 24.2** describes the dedicated GTM implementation this specification is written for. It gives an overview about the implemented submodules.

Section 24.3 up to **Section 24.4** deal with the so called infrastructural components for routing, clock management and common time base functions. **Section 24.5** to **Section 24.7** describe the signal input and output modules. **Section 24.8** describes a module that bundles several interrupts coming from the other submodules and connect them to the outside world.

These submodule groups are shown in the following table:

Table 24-1 Submodule groups

Section	Submodule	Group
Section 24.3	Clock Management Unit (CMU)	Infrastructural components
Section 24.4	Time Base Unit (TBU)	Infrastructural components
Section 24.5	Timer Input Module (TIM)	IO Modules
Section 24.6	Timer Output Module (TOM)	IO Modules

Table 24-1 Submodule groups (cont'd)

Section	Submodule	Group
Section 24.7	Dead Time Module (DTM)	IO Modules
Section 24.8	Interrupt Concentrator Module (ICM)	Interrupt services

24.2 GTM Architecture

24.2.1 Overview

As already mentioned in **Section 24.1** the GTM forms a generic timer platform that serves different application domains and different classes within these application domains. Depending on these multiple requirements of application domains multiple device configurations with different count of submodules (i.e.TIM, TOM, DTM) are possible. In this section the GTM realization is outlined. The architecture of the GTM is depicted in.

24.2.1.1 GTM Architecture Block Diagram

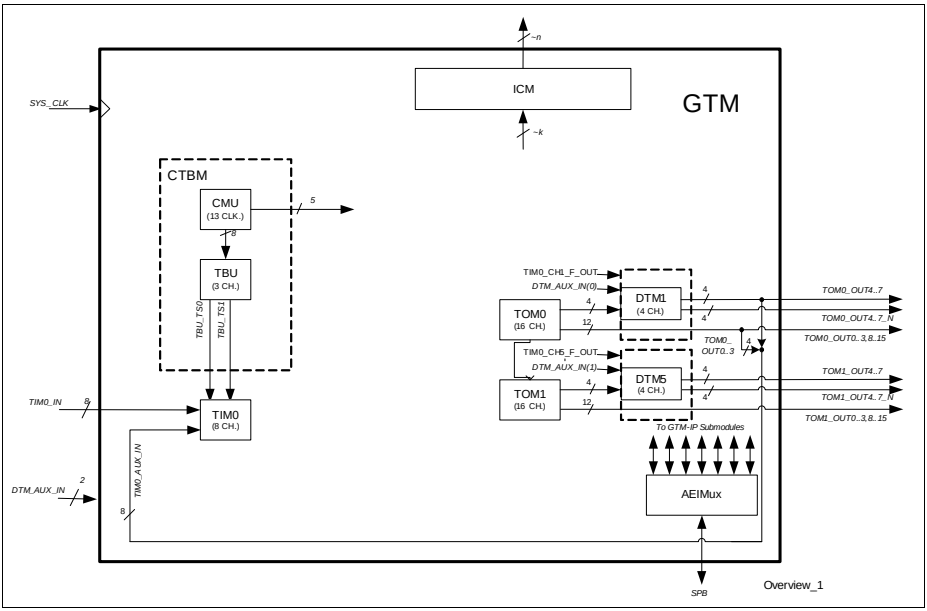


Figure 24-1 GTM Architecture Block Diagram

Signals are transferred into the GTM at the Timer Input Modules (TIM). These modules are able to filter the input signals and annotate additional information. Each channel is for example able to measure pulse high or low times and the period of a PWM signal in parallel. The internal operation registers of the TIM submodule are 24 bits wide.

The Clock Management Unit (CMU) serves up to 13 different clocks for the GTM and up to three external clock pins GTM_ECLK0…2. It acts as a clock divider for the system clock. The counters implemented inside other submodules are typically driven from this submodule. Please note, that the CMU clocks are implemented as enable signals for the counters while the whole system runs with the GTM global clock SYS_CLK. This global clock typically corresponds to the microcontroller bus clock the GTM is connected to.

The TBU provides up to three independent common time bases for the GTM.

Signal outputs are generated with the Dead Time Module (DTM) and Timer Output Modules (TOM). Each TOM channel is able to generate a PWM signal at its output. Because of the integrated shadow register even the generation of complex PWM outputs is possible with the TOM channels by serving the parameters with the CPU. It is possible to trigger TOM channels for a successor TOM submodule through a trigger line between TOM(x)_CH(15) and TOM(x+1)_CH(0). But to avoid long trigger paths the trigger signal is registered at the TOM submodule output, resulting in one SYS_CLK cycle delay of the trigger signal.

In the described implementation the submodules of the GTM have a huge amount of different interrupt sources. These interrupt sources are grouped and concentrated by the Interrupt Concentrator Module (ICM) to form a much easier management bunch of interrupts that are visible outside of the GTM.

On the GTM top level there are some configurable signal connections from the signal output of the DTM modules to the input signals of the TIM modules.

24.2.1.2 GTM signal multiplex

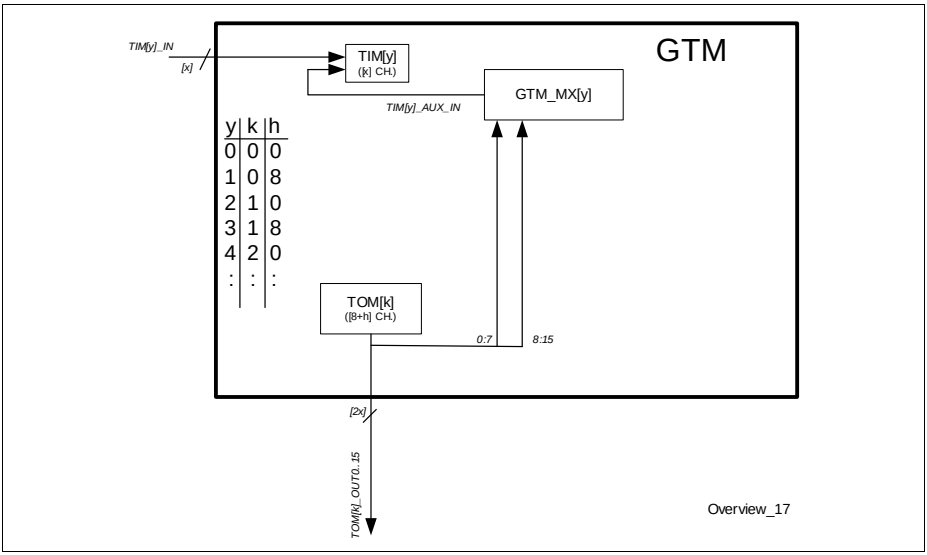


Figure 24-2 GTM Signal Multiplexer

The next diagram gives an overview of the connectivity. The source selection is defined with the bit SRXx in the register GTM_TIM0_AUX_IN_SRC.

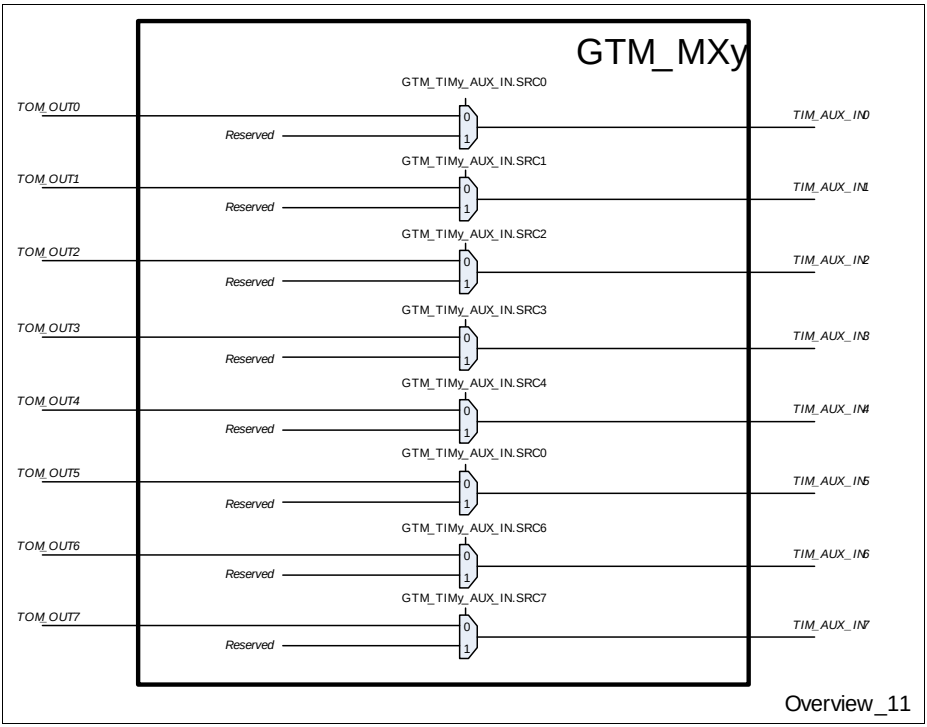


Figure 24-3 GTM TOM to TIM Multiplexer

the trigger out of TIM (i.e. the signals TIM[i]_EXT_CAPTURE(7:0) of each TIMi) are routed to TOM instance [i] with i=0...1.

This TIM trigger can be used to trigger inside TOMi either a channel or a global control register of AGC or TGC0/TGC1 unit.

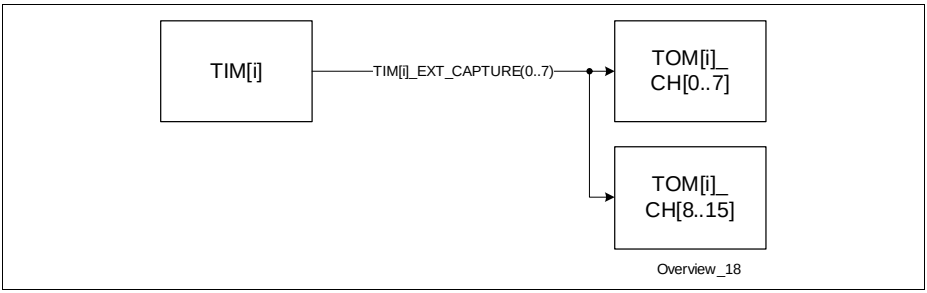


Figure 24-4 TIM to TOM triggers

24.2.2 GTM Interfaces

In general the GTM can be divided into four interface groups. Two interface groups represent the ports of the GTM where incoming signals are assembled and outgoing signals are created. These interfaces are therefore connected to the GTM input submodule TIM and to the GTM output submodules DTM.

Another interface is the bus interface where the GTM can be connected to the SoC system bus. This generic bus interface is described in more detail in **Section 24.2.2.1**. The last interface is the interrupt controller interface. The GTM provides several interrupt lines coming from the various submodules. These interrupt lines are concentrated inside the ICM and have to be adapted to the dedicated microcontroller environment where each interrupt handling can look different. The interrupt concept is described in more detail in **Section 24.2.4**.

24.2.2.1 GTM Generic Bus Interface (AEI)

The GTM is equipped with a generic bus interface that can be widely adapted to different SoC bus systems. This generic bus interface is called AE-Interface (AEI). The adaptation of the AEI to SoC buses is done with a bridge module translating the AEI signals to the SPB bus signals. The AEI bus signals are depicted in the following table:

Table 24-2 AEI Bus Signals

Signal name	I/O	Description	Bit width
AEI_SEL	I	GTM select line	1
AEI_ADDR	I	GTM address	32
AEI_PIPE	I	AEI Address phase signal	1
AEI_W1R0	I	Read/Write access	1
AEI_WDATA	I	Write data bus	32
AEI_RDATA	O	Read data bus	32
AEI_READY	O	Data ready signal	1
AEI_STATUS	O	AEI Access status	2

The AEI Status Signal may drive on of the following values:

Table 24-3 AEI Status Options

AEI_STATUS	Description
00	No Error
01	Illegal Byte Addressing

Table 24-3 AEI Status Options (cont'd)

AEI_STATUS	Description
10	Illegal Address Access
11	Unsupported Address

The signal value "00" is driven if no error occurred during AEI access.

The signal value "01" is driven if the bus address is not an integer multiple of 4 (byte addressing).

The signal value "11" is driven if the address is not handled in the GTM.

The signal value "10" is driven if an illegal write access to one of the following protected register is performed (e.g. protected by bit RF_PROT). Read only register return the status “00”.

Write access to following addresses returns status “10” under special conditions:

CMP_IRQ_FORCINT

CMU_GCLK_NUM

CMU_GCLK_DEN

CMU_CLKx_CTRL (x = 0…7)

CMU_ECLK_NUM

CMU_ECLK_DEN

CMU_FXCLK_CTRL

GTM_IRQ_FORCINT

GTM_RST

TIM0_CHx_CNTS

TIM0_CHx-GPR1

TIM0_CHx_IRQ_FORCINT

TOMi_CHx_IRQ_FORCINT

TBU_CHx_BASE

TBU_CHx_CTRL

24.2.2.2 GTM Multi-master and multi-tasking support

To support multi-master and multi-task access to the registers of the GTM a dedicated write-access scheme is used for critical control bits inside the GTM that need such a mechanism. This can be for example a shared register where more than one channel can be controlled globally by one register write access. Such register bits are implemented inside the GTM with a double bit mechanism, where the writing of '00' and '11' has no effect on the register bit and where '01' sets the bit and '10' resets the bit. If

the CPU wants to read the status of the bit it always gets a '00' if the bit is reset and it gets an '11' if the bit is set.

Note: CPU accesses have to consider the configured SPB to SYS_CLK ratio if SPB > SYSCLK.

24.2.3 GTM Clock and Time Base Management (CTBM)

Inside the GTM several subunits are involved in the clock and time base management of the whole GTM. **Figure 24-5** shows the connections and subblocks involved in these tasks. The subblocks involved are called Clock and Time Base Management (CTBM) modules further on.

24.2.3.1 GTM Clock and time base management architecture

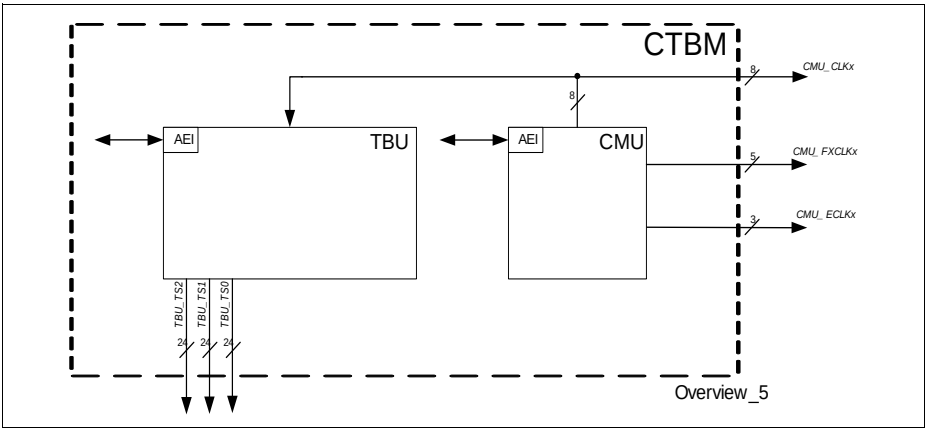


Figure 24-5 GTM Clock and time base management architecture

One important module of the CTBM is the Clock Management Unit (CMU) which generates 13 clocks for the submodules of the GTM and up to three GTM external clocks CMU_ECLK[z] (z: 0…2). For a detailed description of the CMU functionality and clocks please refer to **Section 24.3**.

The five (5) CMU_FXCLK[y] (y: 0…4) clocks are used by the TOM submodule for PWM generation. The eight (8) CMU_CLK[x] (x: 0…7) clocks are used by other submodules of the GTM for signal generation.

Inside the Time Base Unit (TBU) one of these eight clocks is used per channel to generate a common time base for the GTM. The TBU functionality is described in **Section 24.4**.

The TBU submodule generates the three time base signals TBU_TS0, TBU_TS1 and TBU_TS2 which are widely used inside the GTM as common time bases for signal characterization and generation.

24.2.4 GTM Interrupt Concept

The submodules of the GTM can generate many interrupts on behalf of internal events. This high amount of interrupts is combined inside the Interrupt Concentrator Module (ICM) into interrupt groups. In this interrupt groups the GTM submodule interrupt signals are bundled to a smaller set of interrupts. Out of these interrupt sets a smaller amount of interrupt signals is created and signalled outside of the GTM as a signal GTM_<MOD>_IRQ, where as <MOD> identifies the name of the corresponding GTM submodule.

The controlling of the individual interrupts is done inside the submodules. If a submodule consists of several submodule channels that are most likely to work independent from each other (like TIM and TOM), each submodule channel has its own interrupt control and status register set, named as interrupt set in the following. The global GTM functionality have a common interrupt set.

The interrupt set consists of four registers: The IRQ_EN register, the IRQ_NOTIFY register, the IRQ_FORCINT register, and the IRQ_MODE register. While the registers IRQ_EN, IRQ_NOTIFY, and IRQ_FORCINT signalize the status and allow controlling of each individual interrupt source within an interrupt set, the register IRQ_MODE configures the interrupt mode that is applied to all interrupts that belong to the same interrupt set.

In order to support a wide variety of microcontroller architectures and interrupt systems with different interrupt signal output characteristics and internal interrupt handling the following four modes can be configured:

Level mode

Pulse mode

Pulse-Notify mode

Single-Pulse mode

It is recommended to use the Pulse-Notify mode as interrupt mode.

These interrupt modes are described in more details in the following subsections.

The register IRQ_EN allows the enabling and disabling of an individual interrupt within an interrupt set. Independent of the configured mode, only enabled interrupts can signalize an interrupts on its signal GTM_<MOD>_IRQ.

The register IRQ_NOTIFY collects the occurrence of interrupt events.The behaviour for setting a bit in this register depends on the configured mode and thus it is described later on in the mode descriptions.

Independent of the configured mode any write access with value '1' to a bit in the register IRQ_NOTIFY always clears the corresponding IRQ_NOTIFY bit.

Moreover, the enabling of a disabled interrupt sources with a write access to the register IRQ_EN also clears the corresponding bit in the IRQ_NOTIFY register but only if the error interrupt source EIRQ_EN is disabled. However, if the enabling of a disabled interrupt is simultaneous to an incoming interrupt event, the interrupt event is dominant and the register IRQ_NOTIFY is not cleared.

Additionally, each write access to the register IRQ_MODE, clears all bits in the IRQ_NOTIFY register. It should be notified that the clearing of IRQ_NOTIFY is applied independently of the written data (e.g. no mode change).

Thus, a secure way for reconfiguring the interrupt mode of an interrupt set, is to disable all interrupts of the interrupt set with the register IRQ_EN, define the new interrupt mode by writing register IRQ_MODE, followed by enabling the desired interrupts with the register IRQ_EN.

Thus, a secure way for reconfiguring the interrupt mode of an error interrupt set, is to disable all error interrupts of the error interrupt set with the register EIRQ_EN, define the new interrupt mode by writing register IRQ_MODE, followed by enabling the desired error interrupts with the register EIRQ_EN.

The register IRQ_FORCINT is used by software for triggering individual interrupts with a write access with value '1'. Since a write access to IRQ_FORCINT only generates a single pulse, IRQ_FORCINT is not implemented as a true register and thus any read access to IRQ_FORCINT always results with a value of '0'.

It should be noted, that the mechanism for triggering interrupts with IRQ_FORCINT is globally disabled after reset. It has to be explicitly enabled by clearing the bit RF_PROT in the register GTM_CTRL (see **Section 24.2.8.3**)

For the modules AEI-bridge and TIM, each interrupt may configured to raise instead of the normal interrupt an error interrupt if enabled by the corresponding error interrupt enable bit in register EIRQ_EN.

Note, it is possible for one source to enable the normal interrupt and the error interrupt in parallel. Because of both interrupt clear signals could reset the notify bit this is expected to cause problems in a system and therefore it is strongly recommended to not enable both interrupt types at the same point in time.

Similar to enabling an interrupt, the enabling of a disabled error interrupt source with a write access to the register EIRQ_EN also clears the corresponding bit in the IRQ_NOTIFY register only if the interrupt source is disabled. However, if the enabling of a disabled error interrupt is simultaneous to an incoming interrupt event, the interrupt event is dominant and the register IRQ_NOTIFY is not cleared.

All enabled error interrupts are or-combined inside the ICM and assigned to the dedicated GTM port gtm_err_irq.

To be able to detect the module source of the error interrupt the ICM provides the register ICM_IRQG_MEI. The error interrupt causing channel can be determined for the modules TIM by evaluating the ICM register ICM_IRQG_CEI1.

24.2.4.1 Level interrupt mode

The default interrupt mode is the Level Interrupt Mode. In this mode each occurred interrupt event is collected in the register IRQ_NOTIFY, independent of the corresponding enable bit of register IRQ_EN and EIRQ_EN.

An interrupt event, which is defined as a pulse on the signal int_out of **Figure 24-6**, may be triggered by the interrupt source of the submodule or by software performing a write access to the corresponding register IRQ_FORCINT, with a disabled bit RF_PROT in register GTM_CTRL.

Level interrupt mode scheme

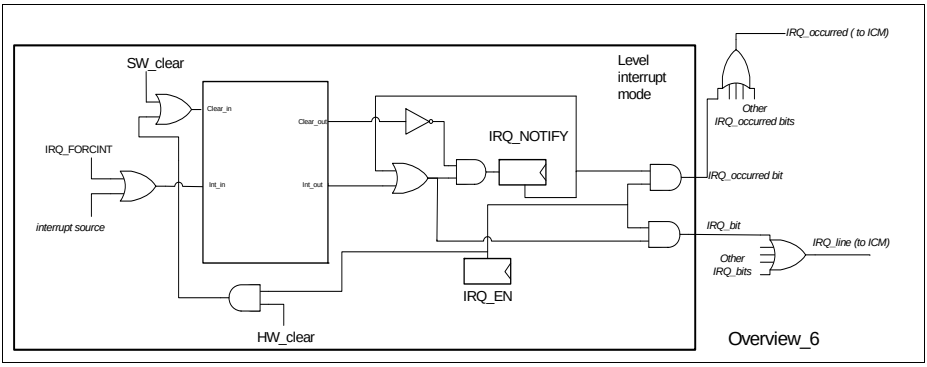


Figure 24-6 Level interrupt mode scheme

A collected interrupt bit in register IRQ_NOTIFY may be cleared by a clear event, which is defined as a pulse on signal clr_out of **Figure 24-6**. A clear event can be performed with a write access with value '1' to the corresponding bit in the register IRQ_NOTIFY leading to a pulse on signals SW_clear. A clear event may also result from an externally connected signal GTM_<MOD>_IRQ_CLR, which is routed to the signal HW_clr of **Figure 24-6**. However, the hardware clear mechanism is only possible, if the corresponding interrupt is enabled by register IRQ_EN.

As the **Table 24-4** shows, interrupt events are dominant in the case of a simultaneous interrupt event and clear event.

Table 24-4 IRQ_NOTIFY behaviour

int_in	clear_in	int_out	clear_out
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	0

Priority of Interrupt Events and Clear Events

As it can be seen from the **Figure 24-6** an occurred interrupt event is signaled as a constant signal level with value 1 to the signal IRQ_bit, if the corresponding interrupt is enabled in register IRQ_EN.

The signal IRQ_bit is OR-combined with the neighboring IRQ_bit signals of the same interrupt set and they are routed as a single signal IRQ_line to the interrupt concentrator module (ICM). In some cases (submodules TOM) the ICM may further OR-combine several IRQ_line signals to an outgoing interrupt signal GTM_<MOD>_IRQ. In the other cases the IRQ_line signals are directly connected to the outgoing signals GTM_<MOD>_IRQ. within the submodule ICM.

The signal IRQ_occurred is connected in a similar way as the signal IRQ_line, however this signal is used for monitoring the interrupt state of the register IRQ_NOTIFY in the registers of the ICM.

The additional error interrupt enable mechanism for level interrupt is shown in **Level interrupt scheme for modules AEI-bridge and TIM**.

Level interrupt scheme for modules AEI-bridge and TIM

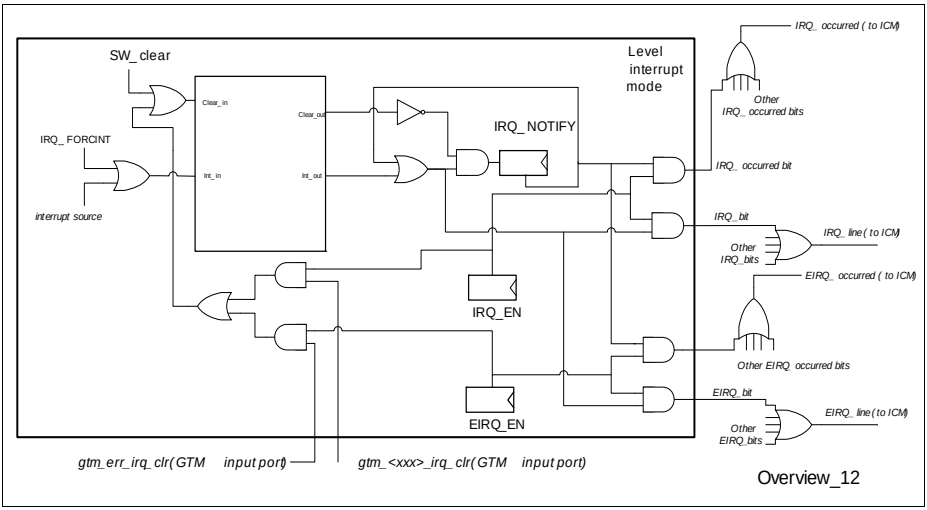


Figure 24-7 Level interrupt scheme

A collected interrupt bit in register IRQ_NOTIFY may be cleared by a clear event, which is defined as a pulse on signal clr_out of **Figure 24-6**. A clear event can be performed with a write access with value '1' to the corresponding bit in the register IRQ_NOTIFY leading to a pulse on signals SW_clear. A clear event may also result from externally connected signal GTM_<MOD>_IRQ_CLR or GTM_ERR_IRQ_CLR, which is routed to the signal HW_clr of **Figure 24-6**. However, the hardware clear mechanism is only possible, if the corresponding interrupt or error interrupt is enabled by register IRQ_EN or EIRQ_EN.

As it can be seen from the **Figure 24-7** an occurred interrupt event is signalled as a constant signal level with value 1 to the signal IRQ_bit, if the corresponding interrupt is enabled in register IRQ_EN.

24.2.4.2 Pulse interrupt mode

The Pulse interrupt mode behaviour can be observed from **Figure 24-8**.

Pulse interrupt mode scheme

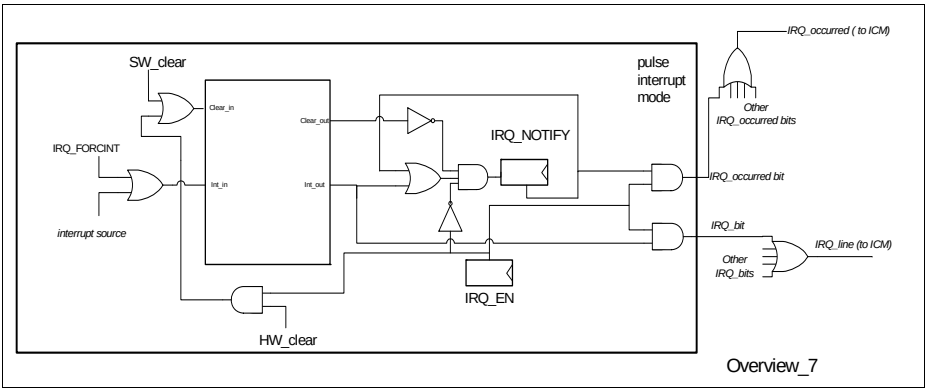


Figure 24-8 Pulse interrupt mode scheme

In Pulse Interrupt Mode each Interrupt Event will generate a pulse on the IRQ_bit signal if IRQ_EN is enabled.

As it can be seen from the figure, the interrupt bit in IRQ_NOTIFY register is always cleared if IRQ_EN or IRQ_EN are enabled.

However, if an interrupt is disabled in the register IRQ_EN, an occurred interrupt event is captured in the register IRQ_NOTIFY, in order to allow polling for disabled interrupts by software.

Disabled interrupts may be cleared by an interrupt clear event.

In Pulse interrupt mode, the signal IRQ_occurred is always 0.

The additional error interrupt enable mechanism for pulse interrupt is shown below.

Pulse interrupt scheme for modules AEI-bridge and TIM

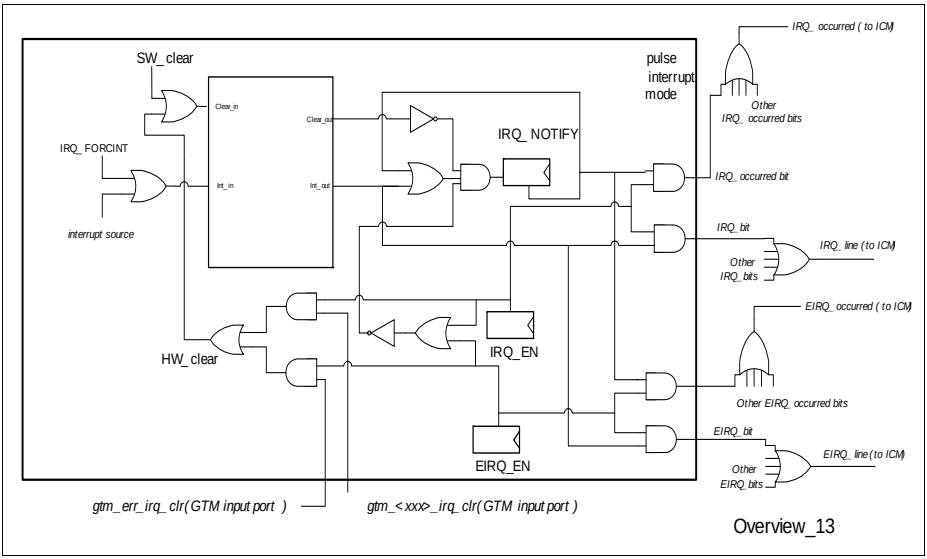


Figure 24-9 Pulse interrupt scheme

In Pulse Interrupt Mode each Interrupt Event will generate a pulse on the EIRQ_bit signal if EIRQ_EN is enabled.

As it can be seen from the figure, the interrupt bit in IRQ_NOTIFY register is always cleared if EIRQ_EN is enabled.

However, if an error interrupt is disabled in the register EIRQ_EN, an occurred error interrupt event is captured in the register IRQ_NOTIFY, in order to allow polling for disabled error interrupts by software.

Disabled error interrupts may be cleared by an error interrupt clear event.

In Pulse interrupt mode, the signal EIRQ_occurred is always 0.

24.2.4.3 Pulse-notify interrupt mode

In Pulse-notify Interrupt mode, all interrupt events are captured in the register IRQ_NOTIFY. If an interrupt is enabled by the register IRQ_EN, each interrupt event will also generate a pulse on the IRQ_bit signal. The signal IRQ_occurred will be high if interrupt is enabled in register IRQ_EN and the corresponding bit of register IRQ_NOTIFY is set. The Pulse-notify interrupt mode is shown in **Figure 24-10**.

Pulse-notify interrupt mode scheme

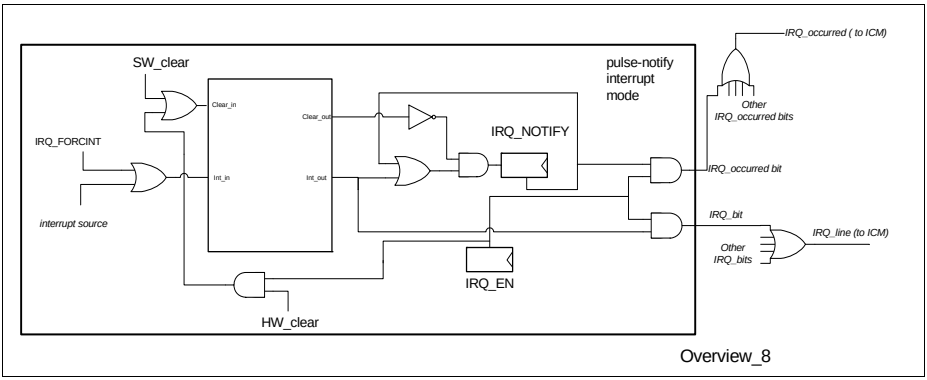


Figure 24-10 Pulse-notify interrupt mode scheme

The additional error interrupt enable mechanism for pulse-notify interrupt is shown below

Pulse-notify interrupt scheme for modules AEI-bridge and TIM

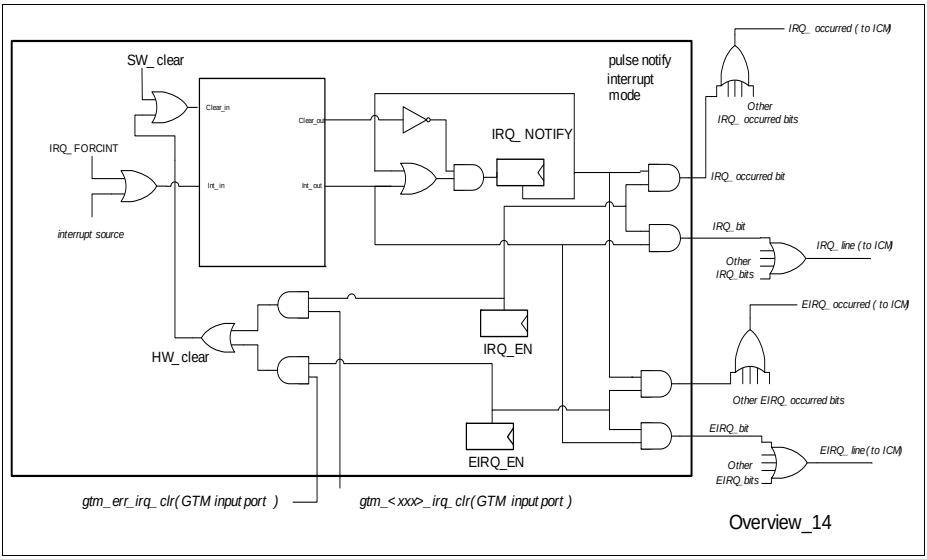


Figure 24-11 Pulse-notify interrupt scheme

In Pulse-notify Interrupt mode, all error interrupt events are captured in the register IRQ_NOTIFY. If an error interrupt is enabled by the register EIRQ_EN, each error

interrupt event will also generate a pulse on the EIRQ_bit signal. The signal EIRQ_occurred will be high if error interrupt is enabled in register EIRQ_EN and the corresponding bit of register IRQ_NOTIFY is set. The Pulse-notify interrupt mode for error interrupts is shown in figure 2.5.3.2.

24.2.4.4 Single-pulse interrupt mode

In Single-pulse Interrupt Mode, an interrupt event is always captured in the register IRQ_NOTIFY, independent of the state of IRQ_EN. However, only the first interrupt event of an enabled interrupt within an common interrupt set is forwarded to signal IRQ_line. Additional interrupt events of the same interrupt set cannot generate pulses on the signal IRQ_line, until the corresponding bits in register IRQ_NOTIFY of enabled interrupts are cleared by a clear event. The IRQ_occurred signal line will be high, if the IRQ_EN and the IRQ_NOTIFY register bits are set. The Single-pulse interrupt mode is shown in **Figure 24-12**.

Single-pulse interrupt mode scheme

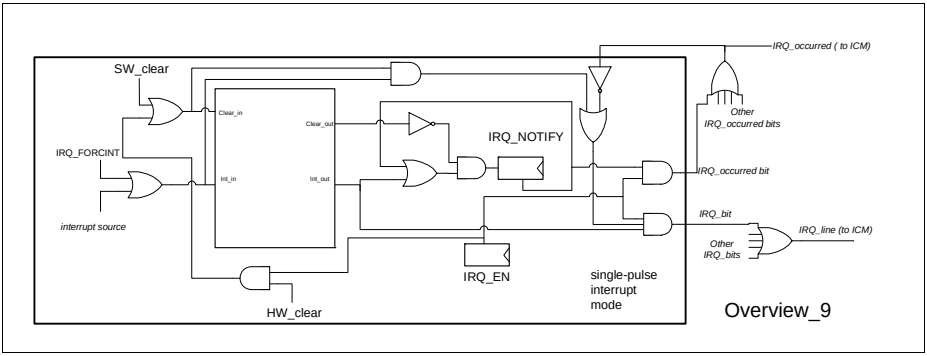


Figure 24-12 Single-pulse interrupt mode scheme

To avoid unexpected IRQ behaviour in the single pulse mode, all desired interrupt sources should be enabled by a single write access to IRQ_EN and the notification bits should be cleared by a single write access to the register IRQ_NOTIFY.

The additional error interrupt enable mechanism for single-pulse interrupt is shown below

Single-pulse interrupt scheme for modules AEI-bridge and TIM

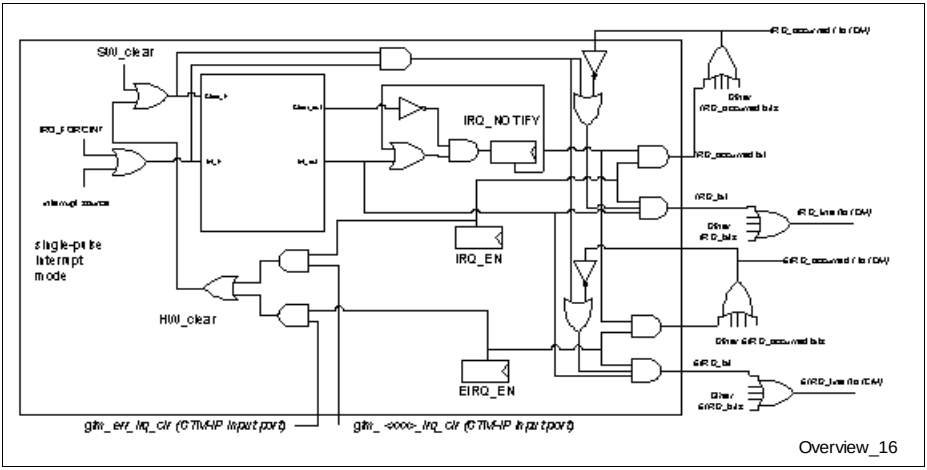


Figure 24-13 Single-pulse interrupt mode scheme

In Single-pulse Interrupt Mode, an error interrupt event is always captured in the register IRQ_NOTIFY, independent of the state of EIRQ_EN. However, only the first error interrupt event of an enabled error interrupt within a common error interrupt set is forwarded to signal EIRQ_line. Additional error interrupt events of the same error interrupt set cannot generate pulses on the signal EIRQ_line, until the corresponding bits in register IRQ_NOTIFY of enabled error interrupts are cleared by a clear event. The EIRQ_occurred signal line will be high, if the EIRQ_EN and the IRQ_NOTIFY register bits are set. The Single-pulse interrupt mode for error interrupts is shown in figure 2.5.4.2.

To avoid unexpected EIRQ behaviour in the single pulse mode, all desired error interrupt sources should be enabled by a single write access to EIRQ_EN and the notification bits should be cleared by a single write access to the register IRQ_NOTIFY.

24.2.4.5 GTM Interrupt concentration method

Because of the grouping of interrupts inside the ICM, it can be necessary for the software to access the ICM submodule first to determine the interrupt set that is responsible for an interrupt. A second access to the responsible register IRQ_NOTIFY is then necessary to identify the interrupt source, serve it and to reset the interrupt flag in register IRQ_NOTIFY afterwards. The interrupt flags are never reset by an access to the ICM. For a detailed description of the ICM submodule please refer to **Chapter 24.8**.

24.2.5 GTM Software Debugger Support

For software debugger support the GTM comes with several features. E.g. status register bits must not be altered by a read access from a software debugger. To avoid this behaviour to reset a status register bit by software, the CPU has to write a '1' explicitly to the register bit to reset its content.

The **Table 24-5** describes the behaviour of some GTM registers with special functionality on behalf of read accesses from the AEI bus interface.

24.2.5.1 Register behaviour in case of Software Debugger accesses

Table 24-5 Register behaviour in case of Software Debugger accesses

Module	Register	Description
TIM	TIM0_CHx_GPR0 / 1	The overflow bit is not altered in case of a debugger read access to this register.

Further on, some important states inside the GTM submodule have to be signalled to the outside world, when reached and should for example trigger the software debugger to stop program execution. For this internal state signalling please refer to the GTM module integration guide.

The GTM provides an external signal gtm_halt, which disables clock signal SYS_CLK for debugging purposes. If SYS_CLK is disabled, a connected debugger can read any GTM related register using AEI. Moreover, the debugger can also perform write accesses to several GTM related registers in order to enable advanced debugging features (e.g. modifications of register contents in single step mode).

24.2.6 GTM Programming conventions

To serve different application domains the GTM is a highly configurable module with many configuration modes. In principle the submodules of the GTM are intended to be configured at system startup to fulfil certain functionality for the application domain the microcontroller runs in.

For example, a TIM input channel can be used to monitor an application specific external signal, and this signal has to be filtered. Therefore, the configuration of the TIM channel filter mode will be specific to the external signal characteristic. While it can be necessary to adapt the filter thresholds during runtime an adaptation of the filter mode during runtime is not reasonable. Thus, the change of the filter mode during runtime can lead to an unexpected behaviour.

In general, the programmer has to be careful when reprogramming configuration registers of the GTM submodules during runtime. It is recommended to disable the channels before reconfiguration takes place to avoid unexpected behaviour of the GTM.

24.2.7 GTM TOP-Level Configuration Registers Overview

GTM TOP-level contains following configuration registers:

Table 24-6 GTM TOP-Level Configuration Registers Overview

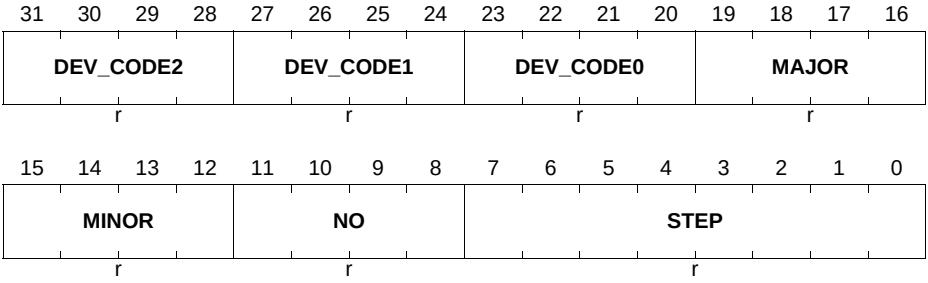
Register name	Description	Details in Section
GTM_REV	GTM Version control register	Section 24.2.8.1
GTM_RST	GTM Global reset register	Section 24.2.8.2
GTM_CTRL	GTM Global control register	Section 24.2.8.3
GTM_AEI_ADDR_XPT	GTM AEI Timeout exception address register	Section 24.2.8.4
GTM_IRQ_NOTIFY	GTM Interrupt notification register	Section 24.2.8.5
GTM_IRQ_EN	GTM Interrupt enable register	Section 24.2.8.6
GTM_IRQ_EIRQ_EN	GTM Error interrupt generation register	Section 24.2.8.12
GTM_IRQ_FORCINT	GTM Software interrupt generation register	Section 24.2.8.7
GTM_IRQ_MODE	GTM top level interrupts mode selection. Please note that this mode selection is only valid for the three interrupts described in Section 24.2.8.5	Section 24.2.8.8
GTM_TIM0_AUX_IN_SRC (i= 0…n)	GTM TIM0 module AUX_IN source selection register	Section 24.2.8.13
GTM_HW_CONF	GTM Hardware Configuration	Section 24.2.8.14

24.2.8 GTM TOP-Level Configuration Registers Description

All of the following registers are 32-bit only accessible.

24.2.8.1 Register GTM_REV

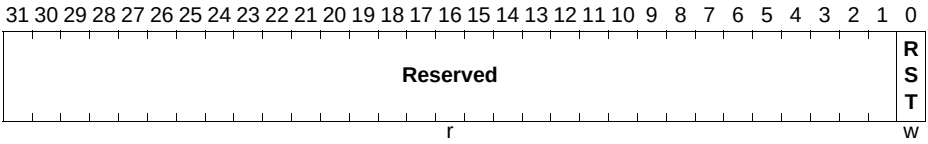
GTM_REV
GTM Version Control Register (00000_H) Reset Value: $210202A1_H$



Field	Bits	Type	Description
STEP	[7:0]	r	**Release Step**
NO	[11:8]	r	**Define delivery number of GTM specification**
MINOR	[15:12]	r	**Define minor version number of GTM specification**
MAJOR	[19:16]	r	**Define major version number of GTM specification**
DEV_CODE0	[23:20]	r	**Device encoding digit 0**
DEV_CODE1	[27:24]	r	**Device encoding digit 1**
DEV_CODE2	[31:28]	r	**Device encoding digit 2** Note: The numbers are encoded in BCD.

24.2.8.2 Register GTM_RST

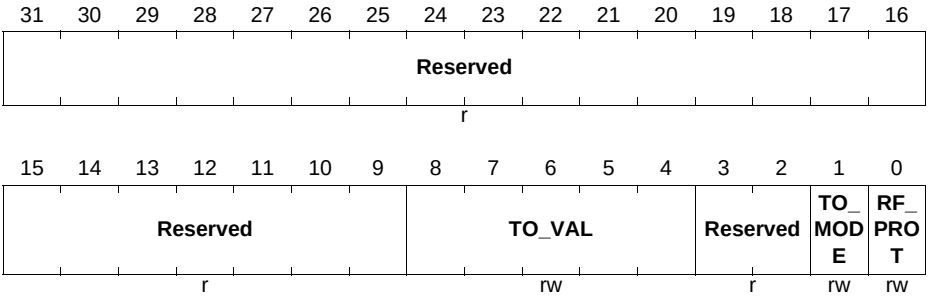
GTM_RST
GTM Global Reset Register (00004_H) Reset Value: 00000000_H



Field	Bits	Type	Description
RST	0	w	**GTM Reset** 0_B No reset action 1_B Initiate reset action for all submodules Note: This bit is automatically cleared by hardware after it was written. Therefore, the register is always read as zero (0) by the software. Note: This bit is write protected by bit GTM_CTRL.RF_PROT.
Reserved	[31:1]	r	**Reserved** Read as zero, should be written as zero

24.2.8.3 Register GTM_CTRL

GTM_CTRL
GTM Global Control Register (00008_H) Reset Value: 00000001_H

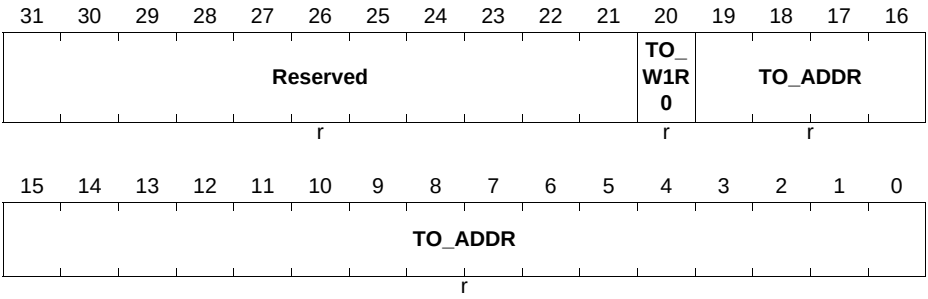


Field	Bits	Type	Description
RF_PROT	0	rw	**RST and FORCINT protection** 0_B SW RST (global), SW interrupt FORCINT, and SW RAM reset functionality is enabled 1_B SW RST (global), SW interrupt FORCINT, and SW RAM reset functionality is disabled
TO_MODE	1	rw	**AEI Timeout mode** 0_B Observe: If timeout_counter=0 the address and rw signal in addition with timeout flag will be stored to the GTM_AEI_ADDR_XPT register. Following timeout_counter=0 accesses will not overwrite the first entry in the aei_addr_timeout register. Clearing the timeout flag/aei_status error_code will reenable the storing of a next faulty access. 1_B Abort: In addition to observe mode the pending access will be aborted by signalling an illegal module access on aei_status and sending ready. In case of a read deliver as data 0 by serving of next AEI accesses.
Reserved	[3:2]	r	**Reserved** Read as zero, should be written as zero

Field	Bits	Type	Description
TO_VAL	[8:4]	rw	**AEI Timeout value** Note: These bits define the number of cycles after which a timeout event occurs. When TO_VAL equals zero (0) the AEI timeout functionality is disabled.
Reserved	[31:9]	r	**Reserved** Read as zero, should be written as zero

24.2.8.4 Register GTM_AEI_ADDR_XPT

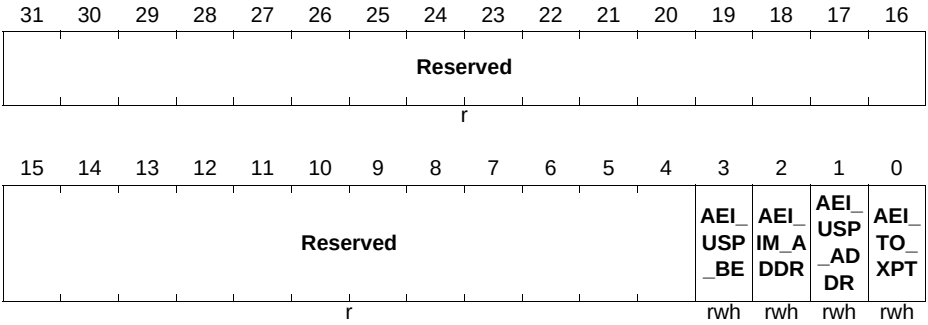
GTM_AEI_ADDR_XPT
GTM AEI Timeout Exception Address Register
(0000C$_H$) **Reset Value: 00000000$_H$**



Field	Bits	Type	Description
TO_ADDR	[19:0]	r	**AEI Timeout address** Note: This bit field defines the AEI address for which the AEI timeout event occurred.
TO_W1R0	20	r	**AEI Timeout Read/Write flag** Note: This bit defines the AEI Read/Write flag for which the AEI timeout event occurred.
Reserved	[31:21]	r	**Reserved** Read as zero, should be written as zero

24.2.8.5 Register GTM_IRQ_NOTIFY

GTM_IRQ_NOTIFY
GTM Interrupt Notification Register (00010_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
AEI_TO_XPT	0	rwh	**AEI Timeout exception occurred** 0_B No interrupt occurred 1_B AEI_TO_XPT interrupt was raised by the AEI Timeout detection unit Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.
AEI_USP_ADDR	1	rwh	**AEI Unsupported address interrupt** 0_B No interrupt occurred 1_B AEI_USP_ADDR interrupt was raised by the AEI interface Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.
AEI_IM_ADDR	2	rwh	**AEI Illegal Module address interrupt** 0_B No interrupt occurred 1_B AEI_IM_ADDR interrupt was raised by the AEI interface Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Field	Bits	Type	Description
AEI_USP_BE	3	rwh	**AEI Unsupported byte enable interrupt** 0_B No interrupt occurred 1_B AEI_USP_BE interrupt was raised by the AEI interface Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.
Reserved	[31:4]	r	**Reserved** Read as zero, should be written as zero

24.2.8.6 Register GTM_IRQ_EN

GTM_IRQ_EN
GTM Interrupt Enable Register (00014_H) Reset Value: 00000000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												AEI_USP_BE_IRQ_EN	AEI_IM_ADDR_IRQ_EN	AEI_USP_ADDR_IRQ_	AEI_TO_XPT_IRQ_EN
r												rw	rw	rw	rw

Field	Bits	Type	Description
AEI_TO_XPT_IRQ_EN	0	rw	**AEI_TO_XPT_IRQ interrupt enable.** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
AEI_USP_ADDR_IRQ_EN	1	rw	**AEI_USP_ADDR_IRQ interrupt enable.** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM

Field	Bits	Type	Description
AEI_IM_ADDR_IRQ_EN	2	rw	**AEI_IM_ADDR_IRQ interrupt enable.** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
AEI_USP_BE_IRQ_EN	3	rw	**AEI_USP_BE_IRQ interrupt enable.** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
Reserved	[31:4]	r	**Reserved** Read as zero, should be written as zero

24.2.8.7 Register GTM_IRQ_FORCINT

GTM_IRQ_FORCINT
GTM Software Interrupt Generation Register
(00018_H) **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r															

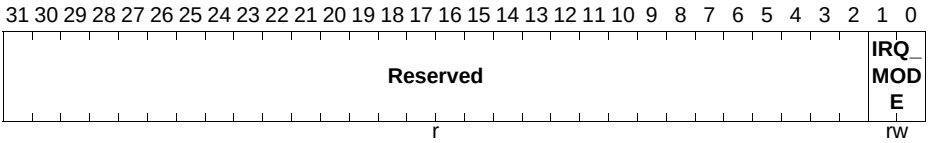
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												**TRG_AEI_USP_BE**	**TRG_AEI_IM_ADDR**	**TRG_AEI_USP_ADDR**	**TRG_AEI_TO_XPT**
r												w	w	w	w

Field	Bits	Type	Description
TRG_AEI_TO_XPT	0	w	**Trigger AEI_TO_XPT_IRQ interrupt by software.** 0_B No interrupt triggering 1_B Assert AEI_TO_XPT_IRQ interrupt for one clock cycle Note: This bit is cleared automatically after write. Note: This bit is write protected by bit GTM_CTRL.RF_PROT.

Field	Bits	Type	Description
TRG_AEI_USP_ADDR	1	w	**Trigger AEI_USP_ADDR_IRQ interrupt by software.** 0_B No interrupt triggering 1_B Assert AEI_USP_ADDR_IRQ interrupt for one clock cycle Note: This bit is cleared automatically after write. Note: This bit is write protected by bit GTM_CTRL.RF_PROT.
TRG_AEI_IM_ADDR	2	w	**Trigger AEI_IM_ADDR_IRQ interrupt by software.** 0_B No interrupt triggering 1_B Assert AEI_IM_ADDR_IRQ interrupt for one clock cycle Note: This bit is cleared automatically after write. Note: This bit is write protected by bit GTM_CTRL.RF_PROT.
TRG_AEI_USP_BE	3	w	**Trigger AEI_USP_BE_IRQ interrupt by software.** 0_B No interrupt triggering 1_B Assert AEI_USP_BE_IRQ interrupt for one clock cycle Note: This bit is cleared automatically after write. Note: This bit is write protected by bit GTM_CTRL.RF_PROT.
Reserved	[31:4]	r	**Reserved** Read as zero, should be written as zero

24.2.8.8 Register GTM_IRQ_MODE

GTM_IRQ_MODE
GTM Top Level Interrupts Mode Selection
($0001C_H$) **Reset Value: 00000000_H**

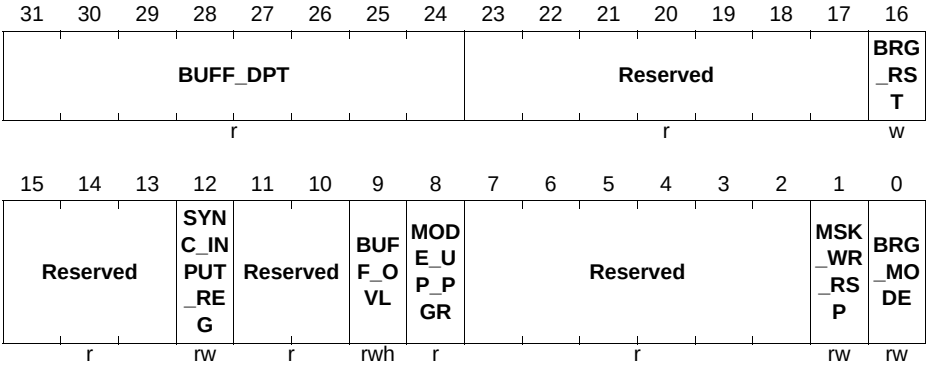


Field	Bits	Type	Description
IRQ_MODE	[1:0]	rw	**Interrupt strategy mode selection for the AEI timeout and address monitoring interrupts** 00_B Level mode 01_B Pulse mode 10_B Pulse-Notify mode 11_B Single-Pulse mode Note: The interrupt modes are described in **Section 24.2.4**.
Reserved	[31:2]	r	**Reserved** Read as zero, should be written as zero

24.2.8.9 Register GTM_BRIDGE_MODE

Note: The content of this register should never be changed.

GTM_BRIDGE_MODE
GTM to SPB BRIDGE MODE **(00030_H)** **Reset Value: 04001001_H**

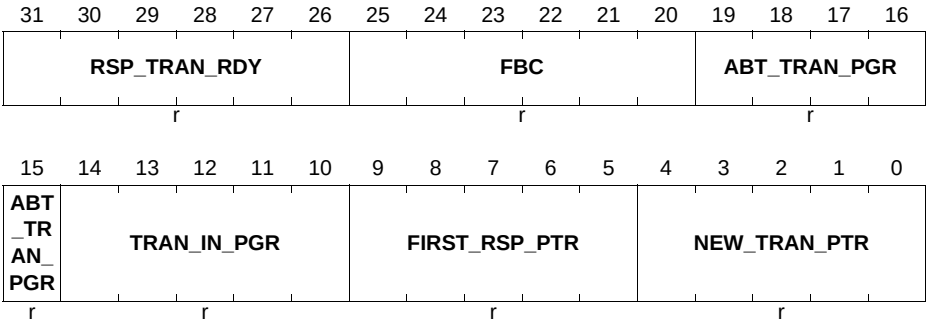


Field	Bits	Type	Description
BRG_MODE	0	rw	**Defines the operation mode for the AEI bridge** 0_B AEI bridge operates in sync_bridge mode 1_B AEI bridge operates in async_bridge mode
MSK_WR_RSP	1	rw	**Mask write response** 0_B Do not mask the write response 1_B Mask write response
Reserved	[7:2]	r	**Reserved** Read as zero, should be written as zero
MODE_UP_PGR	8	r	**Mode update in progress** 0_B No update in progress. 1_B Update in progress.
BUFF_OVL	9	rwh	**Buffer overflow register** 0_B No buffer overflow occurred. 1_B Buffer overflow occurred. Note: A buffer overflow can occur while multiple aborts are issued by the external bus or a pipelined instruction is started while FBC = 0 (see GTM_BRIDGE_PTR1 register). This bit always read as zero.

Field	Bits	Type	Description
Reserved	[11:10]	r	**Reserved** Read as zero, should be written as zero.
SYNC_INPUT_REG	12	rw	**Additional Pipeline Stage in Synchronous Bridge Mode** 0_B No additional pipeline stage implemented 1_B Additional pipeline stage implemented. All accesses in synchronous mode will be increased by one clock cycle.
Reserved	[15:13]	r	**Reserved** Read as zero, should be written as zero.
BRG_RST	16	w	**Bridge software reset** 0_B No bridge reset request. 1_B Bridge reset request. Note: This bit is cleared automatically after write. This bit always read as zero.
BUFF_DPT	[31:24]	r	**Buffer depth of AEI bridge** Signals the buffer depth of the GTM AEI bridge implementation.
Reserved	[23:17]	r	**Reserved** Read as zero, should be written as zero

24.2.8.10 Register GTM_BRIDGE_PTR1

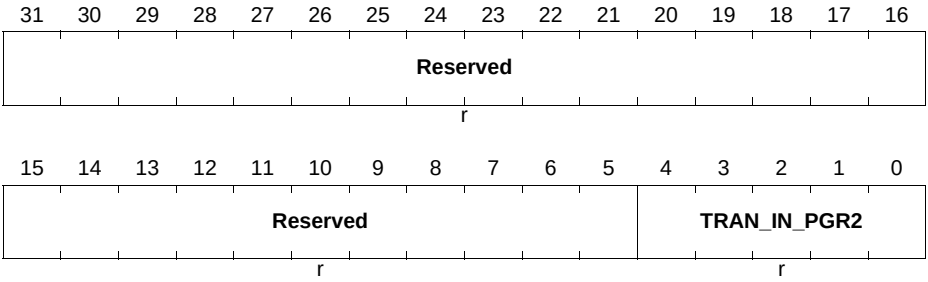
GTM_BRIDGE_PTR1
GTM to SPB BRIDGE PTR1 **(00034_H)** **Reset Value: $00X00000_H$**



Field	Bits	Type	Description
NEW_TRAN_PTR	[4:0]	r	**New transaction pointer** Signals the actual value of the new transaction pointer.
FIRST_RSP_PTR	[9:5]	r	**First response pointer** Signals the actual value of first response pointer.
TRAN_IN_PGR	[14:10]	r	**Transaction in progress pointer (acquire)** Transaction in progress pointer.
ABT_TRAN_PGR	[19:15]	r	**Aborted transaction in progress pointer** Aborted transaction in progress pointer.
FBC	[25:20]	r	**Free buffer count** Number of free buffer entries.
RSP_TRAN_RDY	[31:26]	r	**Response transactions ready** Amount of ready response transactions. Note: This register operates on the AEI_CLK domain.

24.2.8.11 Register GTM_BRIDGE_PTR2

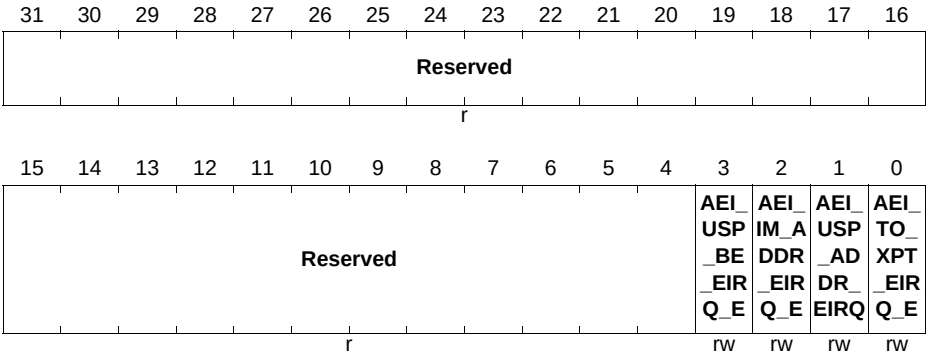
GTM_BRIDGE_PTR2
GTM to SPB BRIDGE PTR2 (00038_H) Reset Value: 00000000_H



Field	Bits	Type	Description
TRAN_IN_PGR2	[4:0]	r	**Transaction in progress pointer (aquire2)** Transaction in progress pointer 2.
Reserved	[31:5]	r	**Reserved** Read as zero, should be written as zero *Note: This register operates on the GTM_CLK domain*

24.2.8.12 Register GTM_EIRQ_EN

GTM_EIRQ_EN
GTM Error Interrupt Enable Register (00020_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
AEI_TO_XPT_EIRQ_EN	0	rw	**AEI_TO_XPT_EIRQ error interrupt enable** 0_B Disable error interrupt, interrupt is not visible outside GTM 1_B Enable error interrupt, interrupt is visible outside GTM
AEI_USP_ADDR_EIRQ_EN	1	rw	**AEI_USP_ADDR_EIRQ error interrupt enable** 0_B Disable error interrupt, interrupt is not visible outside GTM 1_B Enable error interrupt, interrupt is visible outside GTM
AEI_IM_ADDR_EIRQ_EN	2	rw	**AEI_IM_ADDR_EIRQ error interrupt enable** 0_B Disable error interrupt, interrupt is not visible outside GTM 1_B Enable error interrupt, interrupt is visible outside GTM
AEI_USP_BE_EIRQ_EN	3	rw	**AEI_USP_BE_EIRQ error interrupt enable** 0_B Disable error interrupt, interrupt is not visible outside GTM 1_B Enable error interrupt, interrupt is visible outside GTM

Field	Bits	Type	Description
Reserved	[31:4]	r	**Reserved** Read as zero, should be written as zero

24.2.8.13 Register GTM_TIM0_AUX_IN_SRC

GTM_TIM0_AUX_IN_SRC
GTM TIM0 AUX_IN_SRC **(00040_H)** **Reset Value: 00000000_H**

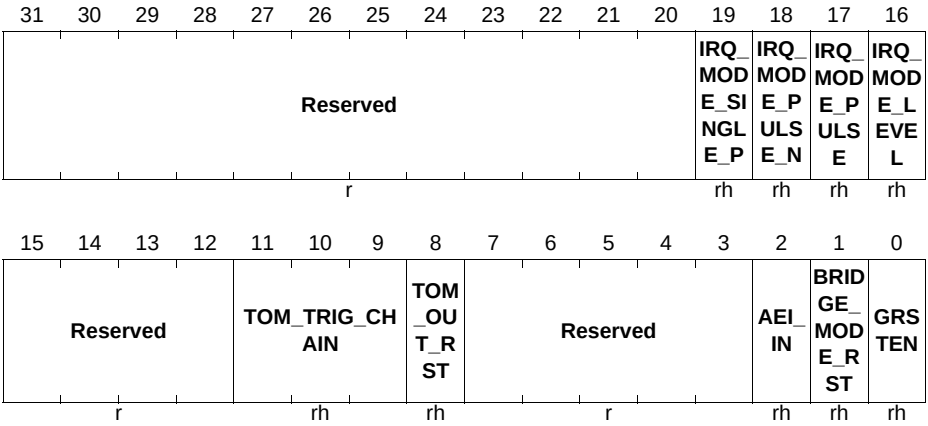
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								SRC_CH7	SRC_CH6	SRC_CH5	SRC_CH4	SRC_CH3	SRC_CH2	SRC_CH1	SRC_CH0
r								rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
SRC_CH0	0	rw	**Defines AUX_IN source of TIM0 channel 0 x=0** 0_B TOM Output selected TOM[a] channel [b] with a= (i* 8 +x) div 16; b=(i* 8 +x) mod 16; 1_B Reserved, do not use this combination
SRC_CH1	1	rw	**Defines AUX_IN source of TIM0 channel 1 x=1, see bit 0**
SRC_CH2	2	rw	**Defines AUX_IN source of TIM0 channel 2 x=2, see bit 0**
SRC_CH3	3	rw	**Defines AUX_IN source of TIM0 channel 3 x=3, see bit 0**
SRC_CH4	4	rw	**Defines AUX_IN source of TIM0 channel 4 x=4, see bit 0**
SRC_CH5	5	rw	**Defines AUX_IN source of TIM0 channel 5 x=5, see bit 0**
SRC_CH6	6	rw	**Defines AUX_IN source of TIM0 channel 6 x=6, see bit 0**
SRC_CH7	7	rw	**Defines AUX_IN source of TIM0 channel 7 x=7, see bit 0**
Reserved	[31:8]	r	**Reserved** Read as zero, should be written as zero

24.2.8.14 Register GTM_HW_CONF

GTM_HW_CONF
GTM Hardware Configuration (00024_H) Reset Value: $000F2247_H$



Field	Bits	Type	Description
GRSTEN	0	rh	**Global Reset Enable** 0_B Global GTM reset register disabled 1_B Global GTM reset register enabled
BRIDGE_MODE_RST	1	rh	**Bridge Mode after Reset** 0_B Bridge starts in synchronous mode after reset 1_B Bridge starts in asynchronous mode after reset
AEI_IN	2	rh	**Input Register in Bridge** 0_B no input register in bridge 1_B input register in bridge
TOM_OUT_RST	8	rh	**TOM_OUT Reset Level** 0_B TOM_OUT reset level is ‘0’ 1_B TOM_OUT reset level is ‘1’ *Note: This reset level defines the reset value of bit SL for all TOM channels*

Field	Bits	Type	Description
TOM_TRIG_CHAIN	[11:9]	rh	**TOM Trigger Chain length without Synchronisation** It defines after which TOM instance count a synchronisation register is introduced into trigger chain (after TOM_TRIG_<i> output if instance i and TOM_TRIG_<i+1> input of instance i+1). Valid values are 1 to 7. 1 means that after each instance a synchronisation register is placed.
IRQ_MODE_LEVEL	16	rh	**IRQ Mode Level** 0_B Level Mode not available 1_B Level Mode available
IRQ_MODE_PULSE	17	rh	**IRQ Mode Pulse** 0_B Pulse Mode not available 1_B Pulse Mode available
IRQ_MODE_PULSE_NOTIFY	18	rh	**IRQ Mode Pulse Notify** 0_B Pulse Notify Mode not available 1_B Pulse Notify Mode available
IRQ_MODE_SINGLE_PULSE	19	rh	**IRQ Mode Single Pulse** 0_B Single Pulse Mode not available 1_B Single Pulse Mode available
Reserved	[7:3], [15:12], [31:20]	r	**Reserved** Read as zero, should be written as zero

24.3 Clock Management Unit (CMU)

24.3.1 Overview

The Clock Management Unit (CMU) is responsible for clock generation of the counters and of the GTM. The CMU consists of three subunits that generate different clock sources for the whole GTM. **Figure 24-14** shows a block diagram of the CMU.

The Configurable Clock Generation (CFGU) subunit provides eight dedicated clock sources for the following GTM submodules: TIM and TBU. Each instance of such a submodule can choose an arbitrary clock source, in order to specify wide-ranging time bases.

The Fixed Clock Generation (FXU) subunit generates predefined non-configurable clocks CMU_FXCLK[y] (y: 0…4) for the TOM submodules. The CMU_FXCLK[y] signals are derived from the CMU_GCLK_EN signal generated by the Global Clock Divider. The dividing factors are defined as 2^0, 2^4, 28, 2^{12}, and 2^{16}.

The External Clock Generation (EGU) subunit is able to generate up to three GTM external clock signals visible at CMU_ECLK[z] (z: 0…2) with a duty cycle of about 50%.

The clock source signals CMU_CLK[x] (x: 0…7) and CMU_FXCLK[y] are implemented in form of enable signals for the corresponding registers, which means that the actual clock signal of all registers always use the SYS_CLK signal.

The four configurable clock signals CMU_CLK0, CMU_CLK1, CMU_CLK6 and CMU_CLK7 are connected to the TIM filter counters.

24.3.1.1 CMU Block Diagram

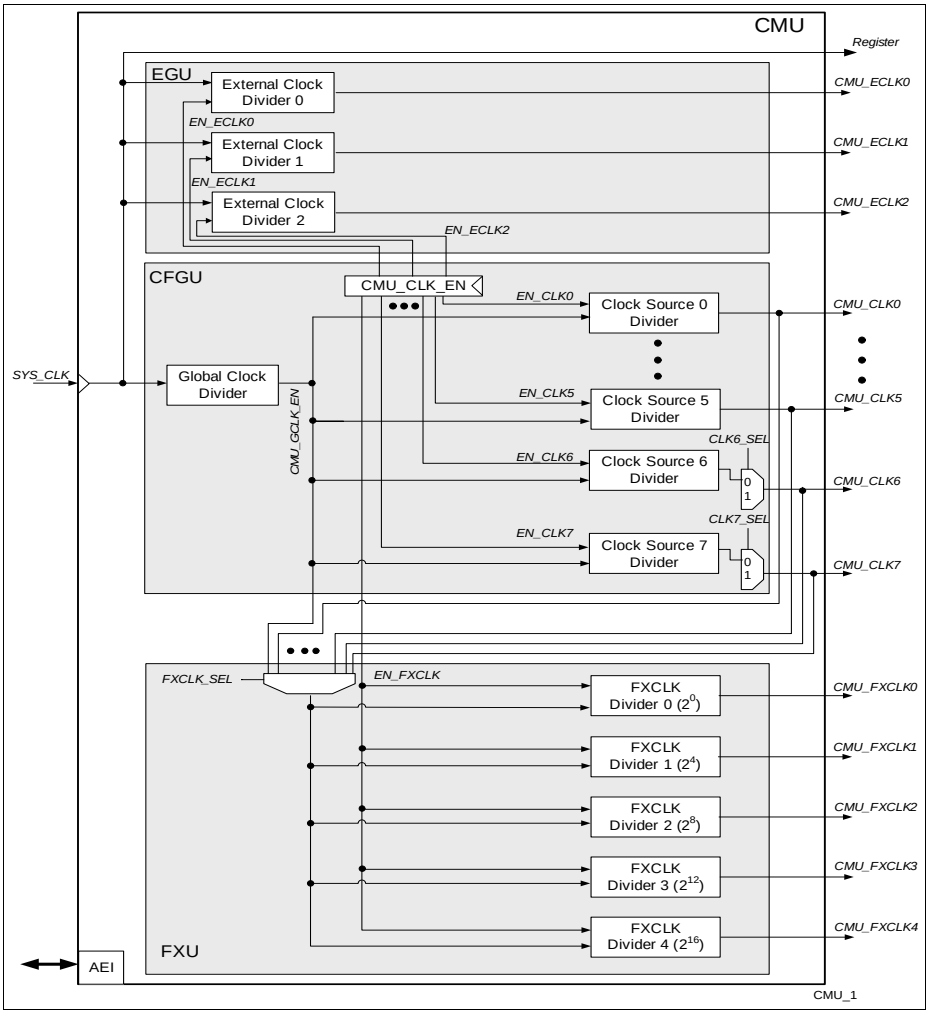


Figure 24-14 CMU Block Diagram

24.3.2 Global Clock Divider

The subblock Global Clock Divider can be used to divide the GTM global input clock signal SYS_CLK into a common subdivided clock signal.

The divided clock signal of the subblock Global Clock Divider is implemented as an enable signal that enables dedicated clocks from the SYS_CLK signal to generate the user specified divided clock frequency.

The resulting fractional divider (Z/N) specified through equation:

TCMU_GCLK_EN=(Z/N)*TSYS_CLK

is implemented according the following algorithm

(Z: CMU_GCLK_NUM(23:0); N: CMU_GCLK_DEN(23:0); Z,N >0):

(1) Set remainder (r), operand1 (OP1) and operand2 (OP2) register during init-phase (with implicit conversion to signed):

r=Z, OP1=N, OP2=N-Z;

(2) After leaving init-phase (at least one CMU_CLK[x] has been enabled) the sign of remainder r for each SYS_CLK cycle will be checked:

(3) If r>0 keep updating remainder and keep CMU_GCLK_EN='0':

r=r-OP1;

(4) If r<0 update remainder and set CMU_GCLK_EN='1':

r=r-OP2;

After at most (Z/N+1) subtractions (3) there will be a negative r and an active phase of the generated clock enable (for one cycle) will be triggered (4). The remainder r is a measure for the distance to a real Z/N clock and will be regarded for the next generated clock enable cycle phase. The new r value will be r=r+(Z-N). In the worst case the remainder r will sum up to an additional cycle in the generated clock enable period after Z-cycles. In the other cases equally distributed additional cycles will be inserted for the generated clock enable. If Z is an integer multiple of N no additional cycles will be included for the generated clock enable at all.

Note that for a better resource sharing all arithmetic has been reduced to subtractions and the initialization of the remainder r uses the complement of (Z-N).

24.3.3 Configurable Clock Generation Subunit (CFGU)

The CMU subunit CFGU provides eight configurable clock divider blocks that divide the common CMU_GCLK_EN signal into dedicated enable signals for the GTM subblocks.

The configuration of the eight different clock signals CMU_CLK[x] (x: 0…7) always depends on the configuration of the global clock enable signal CMU_GCLK_EN. Additionally, each clock source has its own configuration data, provided by the control register CMU_CLK_x_CTRL (x: 0…7).

According to the configuration of the Global Clock Divider, the configuration of the Clock Source x Divider is done by setting an appropriate value in the bit field CLK_CNT[x] of the register CMU_CLK_x_CTRL.

The frequency $f_x = 1/T_x$ of the corresponding clock enable signal CMU_CLK[x] can be determined by the unsigned representation of CLK_CNT[x] of the register CMU_CLK_x_CTRL in the following way:

TCMU_CLK[x]=(CLK_CNT[x]+1)*TCMU_GCLK_EN

The corresponding wave form is shown in **Figure 24-15**.

Each clock signal CMU_CLK can be enabled individually by setting the appropriate bit field EN_CLK[x] in the register CMU_CLK_EN. Except for CMU_CLK6 and CMU_CLK7 individual enabling and disabling is active only if CLK6_SEL and CLK7_SEL is cleared.

To avoid unexpected behaviour of the hardware, the configuration of a register CMU_CLK_x_CTRL can only be changed, when the corresponding clock signal CMU_CLK[x] is disabled.

Further, any changes to the registers CMU_GCLK_NUM and CMU_GCLK_DEN can only be performed, when all clock enable signals CMU_CLK[x] and the EN_FXCLK bit inside the CMU_CLK_EN register are disabled.

The clock source signals CMU_CLK[x] (x:0…7) and CMU_FXCLK[y] are implemented in form of enable signals for the corresponding registers, which means that the actual clock signal of all registers always use the SYS_CLK signal.

The hardware guarantees that all clock signals CMU_CLK[x], which were enabled simultaneous, are synchronized to each other. Simultaneous enabling does mean that the bits EN_CLK[x] in the register CMU_CLK_EN are set by the same write access.

After EN_CLK[x] entries in CMU_CLK_EN have been disabled, internal states of these EN_CLK[x] paths will be reset to start with the same behaviour after enable.

24.3.4 Wave Form of Generated Clock Signal CMU_CLK[x]

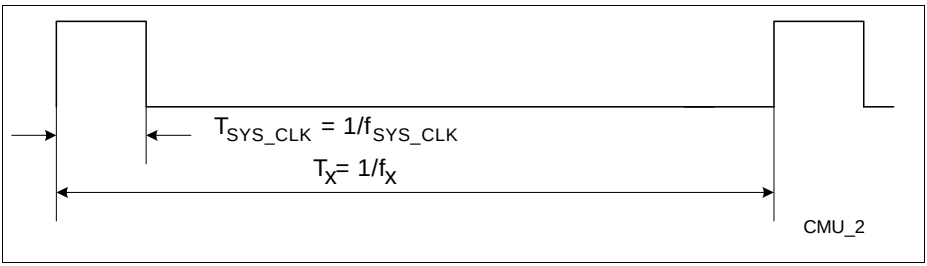


Figure 24-15 Example clock signal

24.3.5 Fixed Clock Generation (FXU)

The FXU subunit generates fixed clock enables out of the CMU_GCLK_EN or one of the eight CMU_CLKx enable signal depending on the CMU_CLK_5.FXCLK_SEL bit field of the CMU_FXCLK_CTRL register. These clock enables are used for the PWM generation inside the TOM submodules.

All clock enables CMU_FXCLK[y] can be enabled or disabled simultaneous by setting the appropriate bit field EN_FXCLK in the register CMU_CLK_EN.

The dividing factors are defined as 2^0, 2^4, 2^8, 2^{12}, and 2^{16}. The signals CMU_FXCLK[y] are implemented in form of enable signals for the corresponding registers (see also **Section 24.3.4**)

After EN_FXCLK entry in CMU_CLK_EN have been disabled, internal states of EN_FXCLK paths will be reset to start with the same behaviour after enable.

24.3.6 External Generation Unit (EGU)

The EGU subunit generate three separate clock output signals CMU_ECLK[z] (z: 0…2).

Each of these clock signals is derived from the corresponding External Clock Divider z subblock, which generates a clock signal derived from the GTM input clock SYS_CLK.

In contrast to the signals CMU_CLK[x] and CMU_FXCLK[y], which are treated as simple enable signals for the registers, the signals CMU_ECLK[z] have a duty cycle of about 50% that is used as a true clock signal for external peripheral components.

Each of the external clocks are enabled and disabled by setting the appropriate bit field EN_ECLK[z] in the register CMU_CLK_EN.

After EN_ECLK[z] entries in CMU_CLK_EN have been disabled, internal states of these EN_ECLK[z] paths will be reset to start with the same behaviour after enable.

The clock frequencies $f_{CMU_ECLK[z]} = 1/T_{CMU_ECLK[z]}$ of the external clocks are controlled with the registers CMU_ECLK_z_NUM and CMU_ECLK_z_DEN as follows:

$$T_{CMU_ECLK[z]} = 2*(ECLK[z]_NUM/ECLK[z]_DEN)*T_{SYS_CLK}$$

and is implemented according the following algorithm

(Z: CMU_ECLK_z_NUM(23:0); N: CMU_ECLK_z_DEN(23:0); Z,N >0; Z>=N; CMU_ECLK[z]='0'):

(1) Set remainder (r), operand1 (OP1) and operand2 (OP2) register during init-phase (with implicit conversion to signed):

r=Z, OP1=N, OP2=N-Z;

(2) After leaving init-phase (CMU_ECLK[z] has been enabled) the sign of remainder r for each SYS_CLK cycle will be checked:

(3) If r>0 keep updating remainder and keep CMU_ECLK[z]:

r=r-OP1;

(4) If r<0 update remainder and toggle CMU_ECLK[z]:

r=r-OP2;

After at most (Z/N+1) subtractions (3) there will be a negative r and an active phase of the generated clock enable (for one cycle) will be triggered (4). The remainder r is a measure for the distance to a real Z/N clock and will be regarded for the next generated clock toggle phase. The new r value will be r=r+(Z-N). In the worst case the remainder r will sum up to an additional cycle in the generated clock toggle period after Z-cycles. In the other cases equally distributed additional cycles will be inserted for the generated clock toggle. If Z is an integer multiple of N no additional cycles will be included for the generated clock toggle at all.

Note that for a better resource sharing all arithmetic has been reduced to subtractions and the initialization of the remainder r uses the complement of (Z-N).

The default value of the CMU_ECLK[z] output is low.

24.3.7 CMU Configuration Registers Overview

Following configuration registers are considered in CMU submodule:

Table 24-7 CMU Configuration Registers Overview

Register Name	Description	Details in Section
CMU_CLK_EN	CMU Clock enable register	Section 24.3.8.1
CMU_GCLK_NUM	CMU Global clock control numerator register	Section 24.3.8.2
CMU_GCLK_DEN	CMU Global clock control denominator register	Section 24.3.8.3
CMU_CLK_x_CTRL	CMU Control for clock source x register (x:0…5)	Section 24.3.8.4
CMU_CLK_6_CTRL	CMU Control for clock source 6 register	Section 24.3.8.5
CMU_CLK_7_CTRL	CMU Control for clock source 7 register	Section 24.3.8.6
CMU_ECLK_z_NUM	CMU External clock z control numerator register (x:0…2)	Section 24.3.8.7
CMU_ECLK_z_DEN	CMU External clock z control denominator register (z:0…2)	Section 24.3.8.8
CMU_FXCLK_CTRL	Control FXCLK subunit input clock	Section 24.3.8.9

24.3.8 CMU Configuration Register Description

All of the following registers are 32-bit only accessible.

24.3.8.1 Register CMU_CLK_EN

GTM_CMU_CLK_EN
CMU Clock Enable Register **(00300_H)** **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								EN_FXCLK		EN_ECLK2		EN_ECLK1		EN_ECLK0	
r								rw		rw		rw		rw	

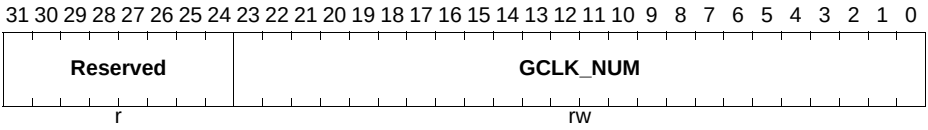
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN_CLK7		EN_CLK6		EN_CLK5		EN_CLK4		EN_CLK3		EN_CLK2		EN_CLK1		EN_CLK0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
EN_CLK0	[1:0]	rw	**Enable clock source 0** 00_B clock source is disabled (ignore write access) 01_B disable clock signal and reset internal states 10_B enable clock signal 11_B clock signal enabled (ignore write access) *Note: Any read access to an EN_CLKx, EN_ECLKz or EN_FXCLK bit field will always result in a value 00 or 11 indicating current state. A modification of the state is only performed with the values 01 and 10. Writing the values 00 and 11 is always ignored.* *Note: Any disabling to EN_CLKx will be reset internal counters for configurable clocks.* *Note: Any disabling to EN_ECLKz will be reset internal counters for external clocks.* *Note: An enable to EN_FXCLK from disable state will be reset internal fixed clock counters.*
EN_CLK1	[3:2]	rw	**Enable clock source 1** see bits [1:0]

Field	Bits	Type	Description
EN_CLK2	[5:4]	rw	**Enable clock source 2** see bits [1:0]
EN_CLK3	[7:6]	rw	**Enable clock source 3** see bits [1:0]
EN_CLK4	[9:8]	rw	**Enable clock source 4** see bits [1:0]
EN_CLK5	[11:10]	rw	**Enable clock source 5** see bits [1:0]
EN_CLK6	[13:12]	rw	**Enable clock source 6** see bits [1:0]
EN_CLK7	[15:14]	rw	**Enable clock source 7** see bits [1:0]
EN_ECLK0	[17:16]	rw	**Enable ECLK 0 generation subunit** see bits [1:0]
EN_ECLK1	[19:18]	rw	**Enable ECLK 1 generation subunit** see bits [1:0]
EN_ECLK2	[21:20]	rw	**Enable ECLK 2 generation subunit** see bits [1:0]
EN_FXCLK	[23:22]	rw	**Enable all CMU_FXCLK** see bits [1:0]
Reserved	[31:24]	r	**Reserved** Read as zero, should be written as zero

24.3.8.2 Register CMU_GCLK_NUM

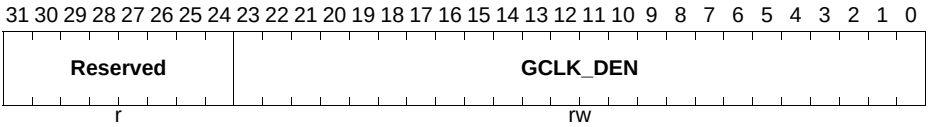
GTM_CMU_GCLK_NUM
CMU Global Clock Control Numerator Register
(00304_H) **Reset Value: 00000001_H**



Field	Bits	Type	Description
GCLK_NUM	[23:0]	rw	**Numerator for global clock divider** Defines numerator of the fractional divider. *Note: Value can only be modified when all clock enables EN_CLK[x] and the EN_FXCLK are disabled.* *Note: The CMU hardware alters the content of CMU_GCLK_NUM and CMU_GCLK_DEN automatically to 0x1, if CMU_GCLK_NUM is specified less than CMU_GCLK_DEN or one of the values is specified with a value zero. Thus, a secure way for altering the values is writing twice to the register CMU_GCLK_NUM followed by a single write to register CMU_GCLK_DEN.*
Reserved	[31:24]	r	**Reserved** Read as zero, should be written as zero

24.3.8.3 Register CMU_GCLK_DEN

GTM_CMU_GCLK_DEN
CMU Global Clock Control Denominator Register
(00308_H) **Reset Value: 00000001_H**

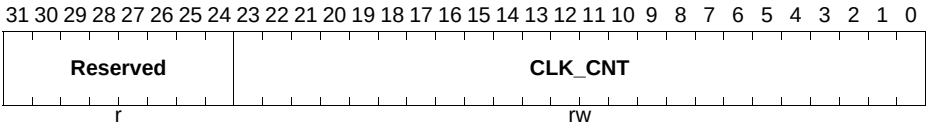


Field	Bits	Type	Description
GCLK_DEN	[23:0]	rw	**Denominator for global clock divider** Defines denominator of the fractional divider. *Note: Value can only be modified when all clock enables EN_CLK[x] and the EN_FXCLK are disabled.* *Note: The CMU hardware alters the content of CMU_GCLK_NUM and CMU_GCLK_DEN automatically to 0x1, if CMU_GCLK_NUM is specified less than CMU_GCLK_DEN or one of the values is specified with a value zero. Thus, a secure way for altering the values is writing twice to the register CMU_GCLK_NUM followed by a single write to register CMU_GCLK_DEN.*
Reserved	[31:24]	r	**Reserved** Read as zero, should be written as zero

24.3.8.4 Register CMU_CLK_x_CTRL (x:0…5)

GTM_CMU_CLK_x_CTRL (x=0-5)
CMU Control For Clock Source x Register

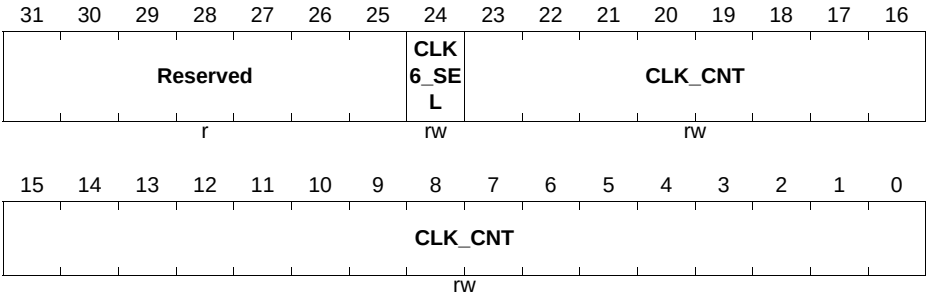
($0030C_H$+x*04_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
CLK_CNT	[23:0]	rw	**Clock count Defines count value for the clock divider of clock source CMU_CLK[x] (x:0…5)** Note: Value can only be modified when clock enable EN_CLKx (x:0…5) is disabled.
Reserved	[31:24]	r	**Reserved** Read as zero, should be written as zero

24.3.8.5 Register CMU_CLK_6_CTRL

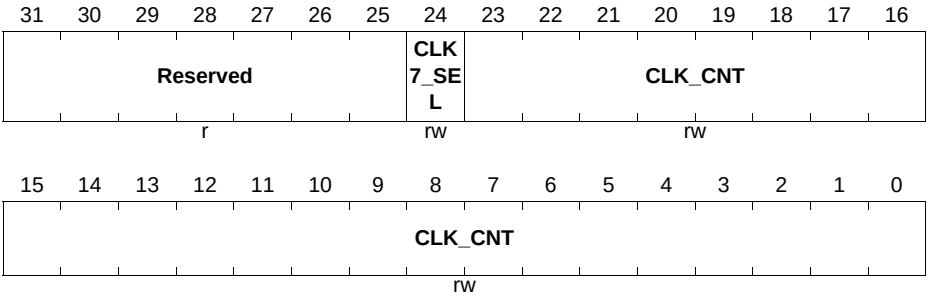
GTM_CMU_CLK_6_CTRL
CMU Control For Clock Source 6 Register
(00324_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
CLK_CNT	[23:0]	rw	**Clock count. Define count value for the clock divider of clock source CMU_CLK6** Note: Value can only be modified when clock enable EN_CLK6 is disabled
CLK6_SEL	24	rw	**Clock source selection for CMU_CLK6** 0_B use Clock Source 6 Divider 1_B Reserved Note: Value can only be modified when clock enable EN_CLK6 is disabled.
Reserved	[31:25]	r	**Reserved** Read as zero, should be written as zero

24.3.8.6 Register CMU_CLK_7_CTRL

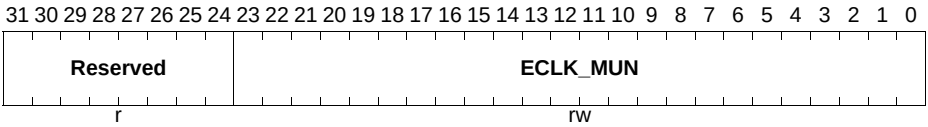
GTM_CMU_CLK_7_CTRL
CMU Control For Clock Source 7 Register
(00328_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
CLK_CNT	[23:0]	rw	**Clock count. Define count value for the clock divider of clock source CMU_CLK7** Note: Value can only be modified when clock enable EN_CLK7 is disabled
CLK7_SEL	24	rw	**Clock source selection for CMU_CLK7** 0_B use Clock Source 7 Divider 1_B Reserved Note: Value can only be modified when clock enable EN_CLK7 is disabled.
Reserved	[31:25]	r	**Reserved** Read as zero, should be written as zero

24.3.8.7 Register CMU_ECLK_z_NUM (z:0…2)

GTM_CMU_ECLK_z_NUM (z=0-2)
CMU External Clock z Control Numerator Register
($0032C_H$+z*08_H) **Reset Value: 00000001_H**

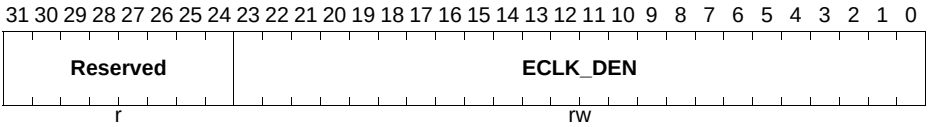


Field	Bits	Type	Description
ECLK_NUM	[23:0]	rw	**Numerator for external clock divider** Defines numerator of the fractional divider. *Note: Value can only be modified when clock enable EN_ECLK[z] is disabled.* *Note: The CMU hardware alters the content of CMU_ECLK_z_NUM and CMU_ECLK_z_DEN automatically to 0x1, if CMU_ECLK_z_NUM is specified less than CMU_ECLK_z_DEN or one of the values is specified with a value zero. Thus, a secure way for altering the values is writing twice to the register CMU_ECLK_z_NUM followed by a single write to register CMU_ECLK_z_DEN.*
Reserved	[31:24]	r	**Reserved** Read as zero, should be written as zero

24.3.8.8 Register CMU_ECLK_z_DEN (z:0…2)

GTM_CMU_ECLK_z_DEN (z=0-2)
CMU External Clock z Control Denominator Register

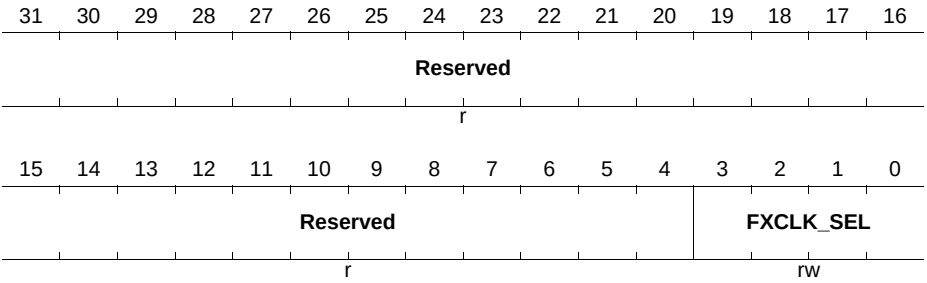
(00330_H + z*08_H) **Reset Value: 00000001_H**



Field	Bits	Type	Description
ECLK_DEN	[23:0]	rw	**Denominator for external clock divider** Defines denominator of the fractional divider. *Note: Value can only be modified when clock enable EN_ECLK[z] is disabled.* *Note: The CMU hardware alters the content of CMU_ECLK_z_NUM and CMU_ECLK_z_DEN automatically to 0x1, if CMU_ECLK_z_NUM is specified less than CMU_ECLK_z_DEN or one of the values is specified with a value zero. Thus, a secure way for altering the values is writing twice to the register CMU_ECLK_z_NUM followed by a single write to register CMU_ECLK_z_DEN.*
Reserved	[31:24]	r	**Reserved** Read as zero, should be written as zero

24.3.8.9 Register CMU_FXCLK_CTRL

GTM_CMU_FXCLK_CTRL
CMU FXCLK Control Register **(00344_H)** **Reset Value: 00000000_H**



Field	Bits	Type	Description
FXCLK_SEL	[3:0]	rw	**Input clock selection for EN_FXCLK line** 0000_BCMU_GCLK_EN selected 0001_BCMU_CLK0 selected 0010_BCMU_CLK1 selected 0011_BCMU_CLK2 selected 0100_BCMU_CLK3 selected 0101_BCMU_CLK4 selected 0110_BCMU_CLK5 selected 0111_BCMU_CLK6 selected 1000_BCMU_CLK7 selected *Note: This value can only be written, when the CMU_FXCLK generation is disabled. See bits 23…22 in register CMU_CLK_EN.* *Note: Other values for FXCLK_SEL are reserved and should not be used, but they behave like FXCLK_SEL = 0.*
Reserved	[31:4]	r	**Reserved bits** Read as zero, should be written as zero

24.4 Time Base Unit (TBU)

24.4.1 Overview

The Time Base Unit TBU provides common time bases for the GTM. The TBU submodule is organized in channels, where the number of channels is device dependent. There are at most three channels implemented inside the TBU. The TBU channel 0 time base register TBU_CH0_BASE is 27 bits and it is configurable whether the lower 24 bit or the upper 24 bit are provided to the GTM as signalTBU_TS0. The two TBU channels 1 and 2 have a time base register TBU_CHy_BASE (y: 1, 2) of 24 bit length. The time base register value TBU_TSy and the time base register update event TBU_UP[y] are provided to subsequent submodules of the GTM.

The TBU_UP[z] (z:1…2) signals are set to high for a single SYS_CLK period, whenever the corresponding signal TBU_TS[z] (z:1…2) or TBU_TS0 is getting updated. The signal TBU_UP0_L is set to high for a single SYS_CLK period if the signal TBU_TS0 and TBU_TS0 is getting updated and TBU_UP0_H is set to high for a single SYS_CLK period, whenever if the upper 24 bit of TBU_TS0 are updated.

The time base channels can run independently of each other and can be enabled and disabled synchronously by control bits in a global TBU channel enable register TBU_CHEN. **Section 24.4.1.1** shows a block diagram of the Time Base Unit.

24.4.1.1 TBU Block Diagram

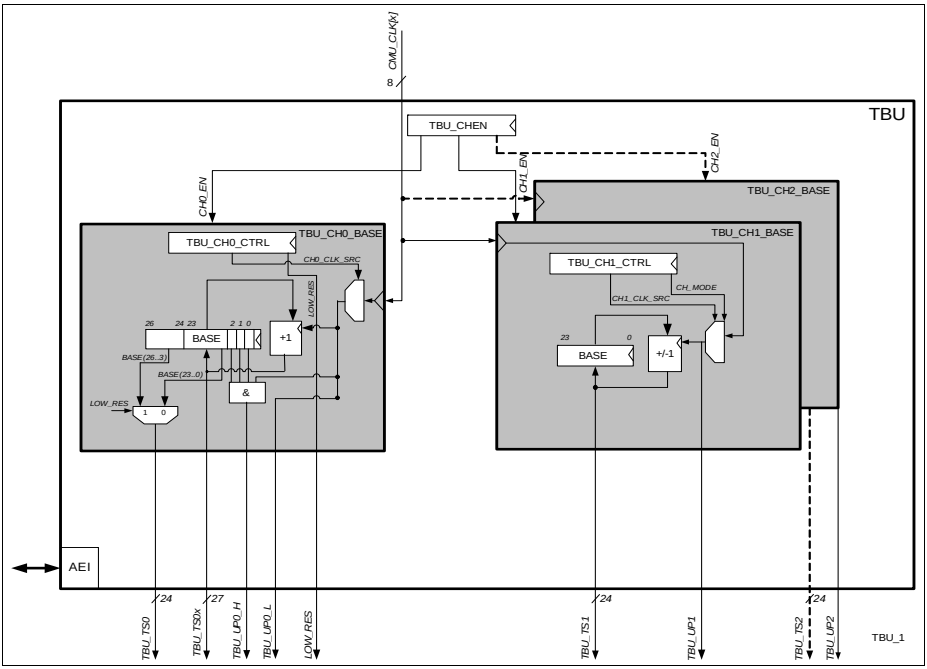


Figure 24-16 TBU Block Diagram

Dependent on the device a third TBU channel exists which offers the same functionality as the time base channel 1.

The configuration of the independent time base channels TBU_BASE_[z] is done via the AEI interface. Each TBU channel may select one of the eight CMU_CLK[x] (x: 0…7) signals coming from the CMU submodule.

24.4.2 TBU Time Base Channels

The time base values are generated within the TBU time base channels in two independent operation modes.

24.4.2.1 TBU Channel Modes

TBU channel 0 provides a 27 bit counter in a free running counter mode. Dependent on the bit field LOW_RES of register TBU_CH0_CTRL, the lower 24 bits (bit 0 to 23) or the upper 24 bits (bits 3 to 26) are provided to the GTM submodules.

TBU channel 1 and channel 2 can run in one mode; the free running counter mode.

In both modes, the time base register TBU_CH[z]_BASE can be initialized with a start value just before enabling the corresponding TBU channel.

Moreover, the time base register TBU_CH[z]_BASE can always be read in order to determine the actual value of the counter.

Free Running Counter Mode

In TBU Free running counter mode, the time base register TBU_CHy_BASE is updated on every specified incoming clock event by the selected signal CMU_CLK[x] (dependent on TBU_CH[z]_CTRL register). In general the time base register TBU_CHy_BASE is increment on every CMU_CLK[x] clock tick.

24.4.3 TBU Configuration Registers Overview

Following table shows a conclusion of configuration registers address offsets and initial values.

Table 24-8 TBU Configuration Registers Overview

Register Name	Description	Details in Section
TBU_CHEN	TBU global channel enable register	Section 24.4.4.1
TBU_CH0_CTRL	TBU channel 0 control register	Section 24.4.4.2
TBU_CH0_BASE	TBU channel 0 base register	Section 24.4.4.3
TBU_CHy_CTRL	TBU channel y control register (y:1,2)	Section 24.4.4.4
TBU_CHy_BASE	TBU channel y base register (y:1,2)	Section 24.4.4.5

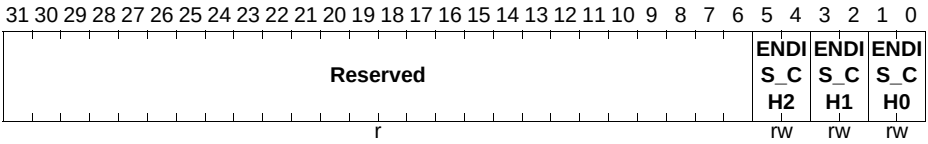
Note: In a typical application the Time Base Unit (TBU) considers channels 0 and 1 only. In this case register addresses 0x20…0x2C are reserved and shall be read as zero. Channel 2 can be additionally implemented on special high-end application requirements.

24.4.4 TBU Registers description

All of the following registers are 32-bit only accessible.

24.4.4.1 Register TBU_CHEN

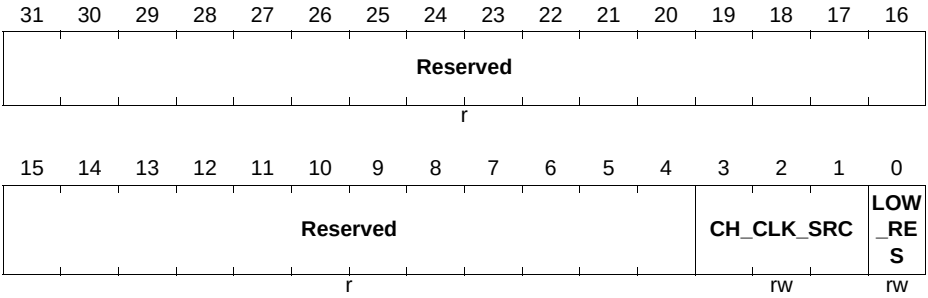
GTM_TBU_CHEN
TBU Global Channel Enable Register(00100_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
ENDIS_CH0	[1:0]	rw	**TBU channel 0 enable/disable control** Write of following double bit values is possible: 00_B don't care, bits 1:0 will not be changed 01_B channel disabled: is read as 00 (see below) 10_B channel enabled: is read as 11 (see below) 11_B don't care, bits 1:0 will not be changed Note: Read of following double values means: 00_B channel disabled 11_B channel enabled
ENDIS_CH1	[3:2]	rw	**TBU channel 1 enable/disable control** See bits 1:0
ENDIS_CH2	[5:4]	rw	**TBU channel 2 enable/disable control** See bits 1:0 Note: These bits are only applicable if channel is implemented for this device, otherwise read and write as zero
Reserved	[31:6]	r	**Reserved** Read as zero, should be written as zero

24.4.4.2 Register TBU_CH0_CTRL

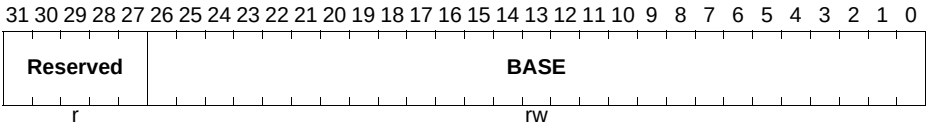
GTM_TBU_CH0_CTRL
TBU Channel 0 Control Register (00104_H) Reset Value: 00000000_H



Field	Bits	Type	Description
LOW_RES	0	rw	**TBU_CH0_BASE register resolution** 0_B TBU channel uses lower counter bits (bit 0 to 23) 1_B TBU channel uses upper counter bits (bit 3 to 26) Note: The two resolutions for the TBU channel 0 can be used in the TIM channel 0 submodule. Note: This value can only be modified if channel 0 is disabled.
CH_CLK_SRC	[3:1]	rw	**Clock source for channel x (x:0…2) time base counter** 000_B CMU_CLK0 selected 001_B CMU_CLK1 selected 010_B CMU_CLK2 selected 011_B CMU_CLK3 selected 100_B CMU_CLK4 selected 101_B CMU_CLK5 selected 110_B CMU_CLK6 selected 111_B CMU_CLK7 selected Note: This value can only be modified (written) if channel 0 was disabled
Reserved	[31:4]	r	**Reserved** Read as zero, should be written as zero

24.4.4.3 Register TBU_CH0_BASE

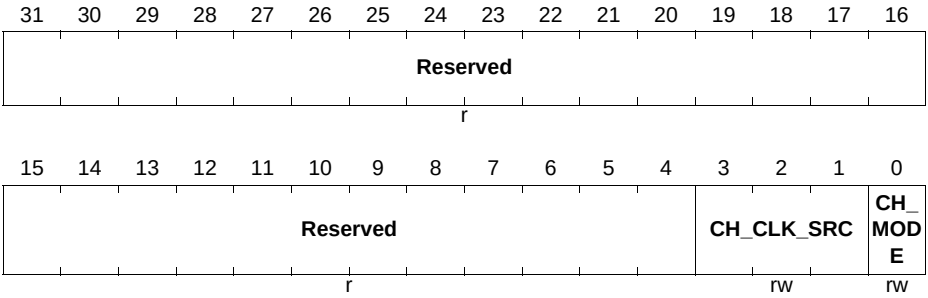
GTM_TBU_CH0_BASE
TBU Channel 0 Base Register (00108_H) Reset Value: 00000000_H



Field	Bits	Type	Description
BASE	[26:0]	rw	**Time base value for channel 0** Note: The value of BASE can only be written if the TBU channel 0 is disabled Note: If channel 0 is enabled, a read access to this register provides the current value of the underlying 27 bit counter.
Reserved	[31:27]	r	**Reserved** Read as zero, should be written as zero

24.4.4.4 Register TBU_CHy_CTRL (y:1, 2)

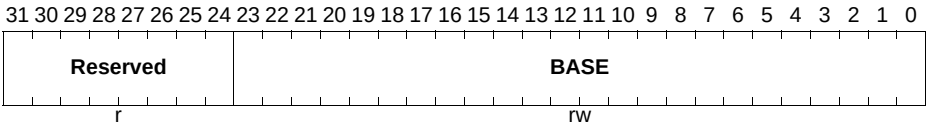
GTM_TBU_CHy_CTRL (y=1-2)
TBU Channel y Control Register (00104_H+y*08_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
CH_MODE	0	rw	**Channel mode** 0_B Free running counter mode 1_B Reserved Note: This value can only be modified if channel y (y:1,2) was disabled. In Free running counter mode the CMU clock source specified by CH_CLK_SRC is used for the counter.
CH_CLK_SRC	[3:1]	rw	**Clock source for channel x (x1…2) time base counter** 000_B CMU_CLK0 selected 001_B CMU_CLK1 selected 010_B CMU_CLK2 selected 011_B CMU_CLK3 selected 100_B CMU_CLK4 selected 101_B CMU_CLK5 selected 110_B CMU_CLK6 selected 111_B CMU_CLK7 selected Note: This value can only be modified if channel y was disabled
Reserved	[31:4]	r	**Reserved** Read as zero, should be written as zero

24.4.4.5 Register TBU_CHy_BASE (y:1,2)

GTM_TBU_CHy_BASE (y=1-2)
TBU Channel y Base Register (00108_H+y*08_H) Reset Value: 00000000_H



Field	Bits	Type	Description
BASE	[23:0]	rw	**Time base value for channel x (x 1, 2)** Note: The value of BASE can only be written if the corresponding TBU channel y is disabled Note: If the corresponding channel y is enabled, a read access to this register provides the current value of the underlying counter.
Reserved	[31:24]	r	**Reserved** Read as zero, should be written as zero

24.5 Timer Input Module (TIM)

24.5.1 Overview

The Timer Input Module (TIM) is responsible for filtering and capturing input signals of the GTM. Several characteristics of the input signals can be measured inside the TIM channels.

Input characteristics mean either time stamp values of detected input rising or falling edges together with the new signal level or the number of edges received since channel enable together with the actual time stamp or PWM signal durations for a whole PWM period.

The architecture of TIM is shown in **Figure 24-17**.

24.5.1.1 TIM Block Diagram

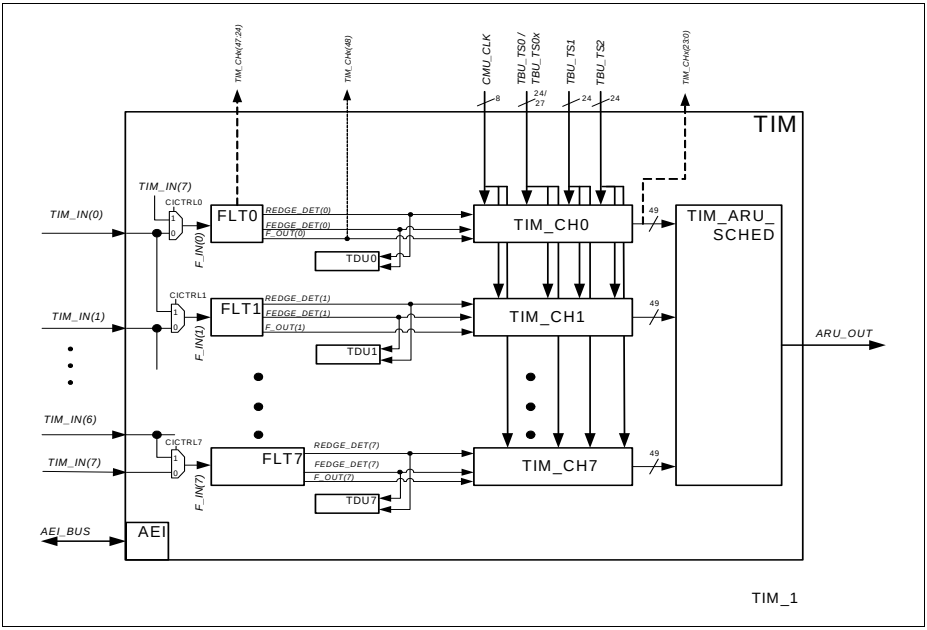


Figure 24-17 TIM Block Diagram

Each of the eight (8) dedicated input signals is filtered inside the FLTx subunit of the TIM Module. It should be noted that the incoming input signals are synchronized to the clock SYS_CLK, resulting in a delay of two SYS_CLK periods for the incoming signals.

The submodule TIM provides different filter mechanisms described in more detail in **Section 24.5.2**. After filtering, the signal is routed to the corresponding TIM channel.

The measurement values can be read by the CPU directly via the AEI-Bus.

For timeout detection of an incoming signal (no subsequent edge detected during a specified duration) each individual channel has a Timeout Detection Unit (TDU).

The two (three) time bases coming from the TBU are connected to the TIM channels to annotate time stamps to incoming signals. For TIM0 the extended 27 bit width time base TBU_TS0 is connected to the TIM channels, and the user has to select if the lower 24 bits (TBU_TS0(23…0)) or the higher 24 bits (TBU_TS0(26…3)) are stored inside the GPRz registers.

24.5.1.2 Input source selection INPUTSRCx

It can be configured which source shall be used for processing in the FLT,TDU,TIM_CH units. It can be selected by the bit fields CICTRL and MODE_x, VAL_x in the register TIM0_CHx_IN_SRC which source is in use.

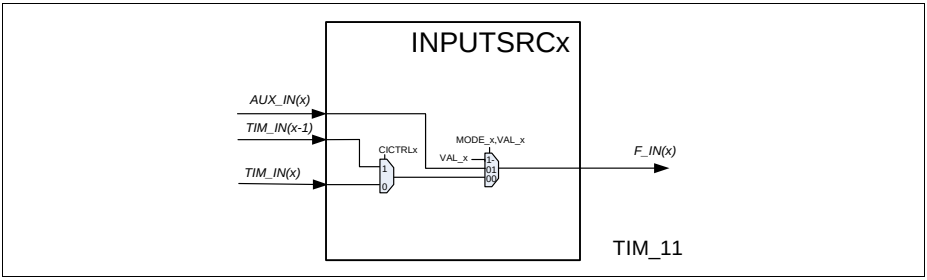


Figure 24-18 TIM Source Selection

In a certain MODE_x, VAL_x combination the input signal F_IN(x) can be driven by VAL_x with 0 or 1 directly.

Due to the fact that all 8 channels are bundled in the register TIM0_CHx_IN_SRC a synchronous control of all 8 input channels is possible.

Two adjacent channels can be combined by setting the CICTRL bit field in the corresponding TIM0_CHx_CTRL register. This allows for a combination of complex measurements on one input signal with two TIM channels.

The additional input signal AUX_IN[x] can be selected as an input signal. The source of this signal is defined in the **Section 24.2.1.2**.

24.5.1.3 Input Observation

It is possible to observe for all channels of one instance by reading TIM_INP_VAL the actual signal values of the following processing stages:

- TIM_IN(7:0) signals after TIM input synchronisation
- TIM F_IN(7:0) signals after TIM INPUTSRC selection (input to TIM_FLT)
- TIM F_OUT(7:0) signals after TIM filter functionality (output of TIM_FLT)

24.5.1.4 External capture source selection EXTCAPSRCx

Each channel can operate on an external capture signal EXT_CAPTURE. The source to use for this signal can be configured by the bit field EXT_CAP_SRCx in the register TIM0_CHx_ECTRL.

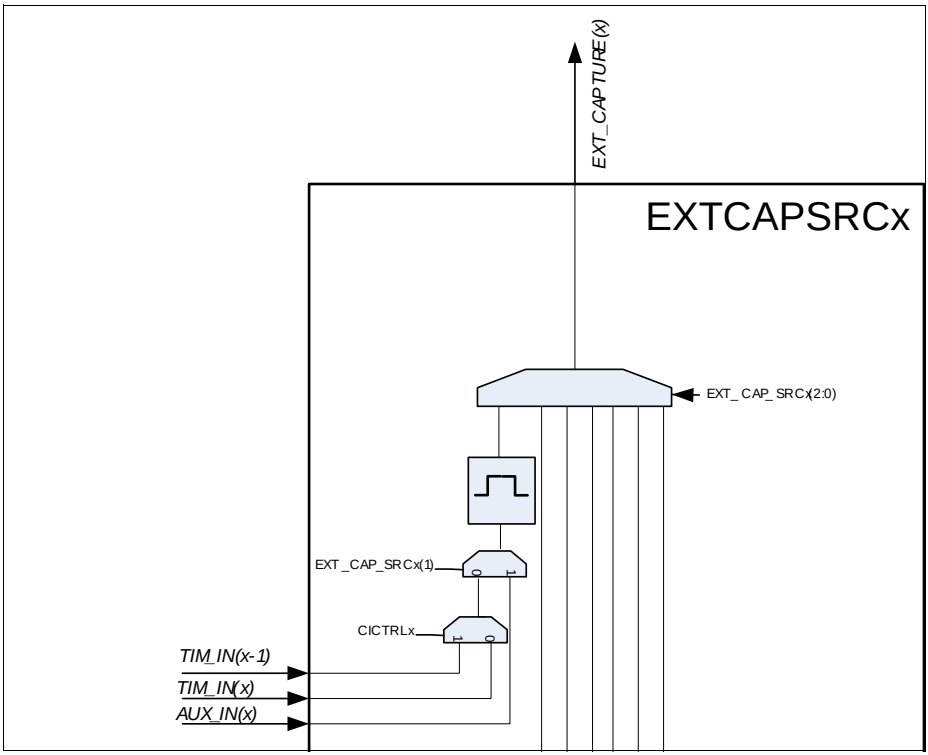


Figure 24-19 TIM external Capture Interrupt

The external capture functionality can be enabled with the bit EXT_CAP_EN

in the register TIM0_CHx_CTRL, it will trigger on each rising edge. A pulse generation for each rising edge of the selected input signal TIM_IN[x] and AUX_IN[x] is applied.

The six TIM channel interrupt sources can be triggered by the operation in the certain TIM channel modes. Alternatively they can be issued by a soft trigger using the corresponding bits in the register TIM0_CHx+1_FORCINT.

24.5.2 TIM Filter Functionality (FLT)

24.5.2.1 Overview

The TIM submodule provides a configurable filter mechanism for each input signal. These filter mechanism is provided inside the FLT subunit.

FLT architecture is shown in **Figure 24-20**

The filter includes a clock synchronisation unit (CSU), an edge detection unit (EDU), and a filter counter associated to the filter unit (FLTU).

The CSU is synchronizing the incoming signal F_IN to the selected filter clock frequency, which is controlled with the bit field FLT_CNT_FRQ of register TIM0_CHx_CTRL.

The synchronized input signal F_IN_SYNC is used for further processing within the filter.

It should be noted that glitches with a duration less than the selected CMU clock period are lost.

The filter modes can be applied individually to the falling and rising edges of an input signal. The following filter modes are available:

- immediate edge propagation mode,
- individual de-glitch time mode (up/down counter), and
- individual de-glitch time mode (hold counter).

FLT Architecture

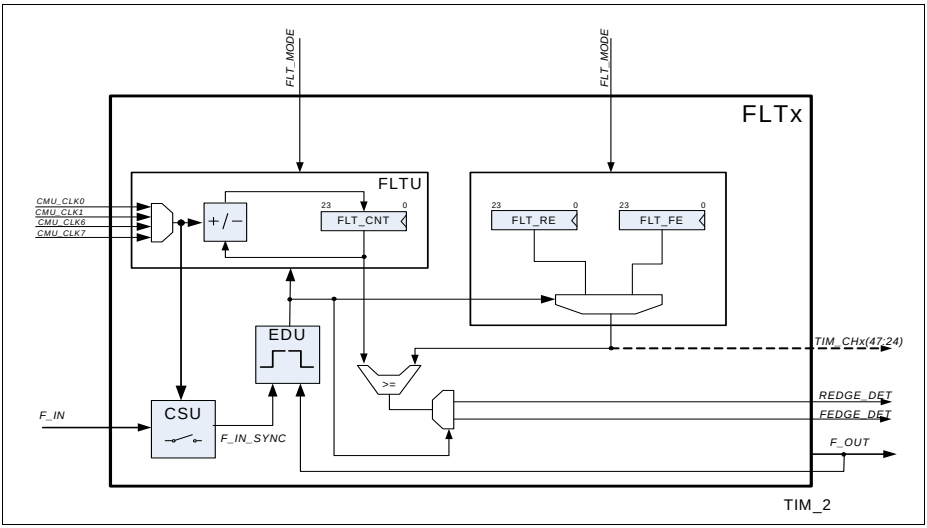


Figure 24-20 FLT Architecture

The filter parameters (deglitch and acceptance time) for the rising and falling edge can be configured inside the two filter parameter registers FLT_RE (rising edge) and FLT_FE (falling edge). The exact meaning of the parameter depends on the filter mode.

However the delay time T of both filter parameters FLT_xE can always be determined by:

T=(FLT_xE+1)*TFLT_CLK,

whereas TFLT_CLK is the clock period of the selected CMU clock signal in bit field FLT_CNT_FRQ of register TIM0_CHx_CTRL.

When a glitch is detected on an input signal a status flag GLITCHDET is set inside the TIM0_CHx_IRQ_NOTIFY register.

Table 24-9 gives an overview about the meanings for the registers FLT_RE and FLT_FE. In the individual deglitch time modes, the actual filter threshold for a detected regular edge is provided on the TIM0_CH[x](47:24) output line. In the case of immediate edge propagation mode, a value of zero is provided on the TIM0_CH[x](47:24) output line.

Filter Parameter summary for the different Filter Modes

Table 24-9 Filter Parameter summary for the different Filter Modes

Filter mode	Meaning of FLT_RE	Meaning of FLT_FE
Immediate edge propagation	Acceptance time for rising edge	Acceptance time for falling edge
Individual de-glitch time (up/down counter)	De-glitch time for rising edge	De-glitch time for falling edge
Individual de-glitch time (hold counter)	De-glitch time for rising edge	De-glitch time for falling edge

A counter FLT_CNT is used to measure the glitch and acceptance times.

The frequency of the FLT_CNT counter is configurable in bit field FLT_CNT_FRQ of register TIM0_CHx_CTRL.

The counter FLT_CNT can either be clock with the CMU_CLK0, CMU_CLK1, CMU_CLK6 or the CMU_CLK7 signal. These signals are coming from the CMU submodule.

The FLT_CNT, FLT_FE and FLT_RE registers are 24-bit width. For example, when the resolution of the CMU_CLK0 signal is 50ns this allows maximal de-glitch and acceptance times of about 838ms for the filter.

24.5.2.2 TIM Filter Modes

Immediate Edge Propagation Mode

In immediate edge propagation mode after detection of an edge the new signal level on F_IN_SYNC is propagated to F_OUT with a delay of one TFLT_CLK period and the new signal level remains unchanged until the configured acceptance time expires.

For each edge type the acceptance time can be specified separately in the FLT_RE and FLT_FE registers.

Each signal change on the input F_IN_SYNC during the duration of the acceptance time has no effect on the output signal level F_OUT of the filter but it sets the glitch GLITCHDET bit in the TIM0_CHx_IRQ_NOTIFY register.

After it expires an acceptance time the input signal F_IN_SYNC is observed and on signal level change the filter raises a new detected edge and the new signal level is propagated to F_OUT.

Independent of a signal level change the value of F_OUT is always set to F_IN_SYNC, when the acceptance time expires (see also **Figure 24-22**).

Figure 24-21 shows an example for the immediate edge propagation mode, in the case of rising edge detection. Both, the signal before filtering (F_IN) and after filtering (F_OUT) are shown. The acceptance time at1 is specified in the register FLT_RE.

Immediate Edge Propagation Mode in the case of a rising edge

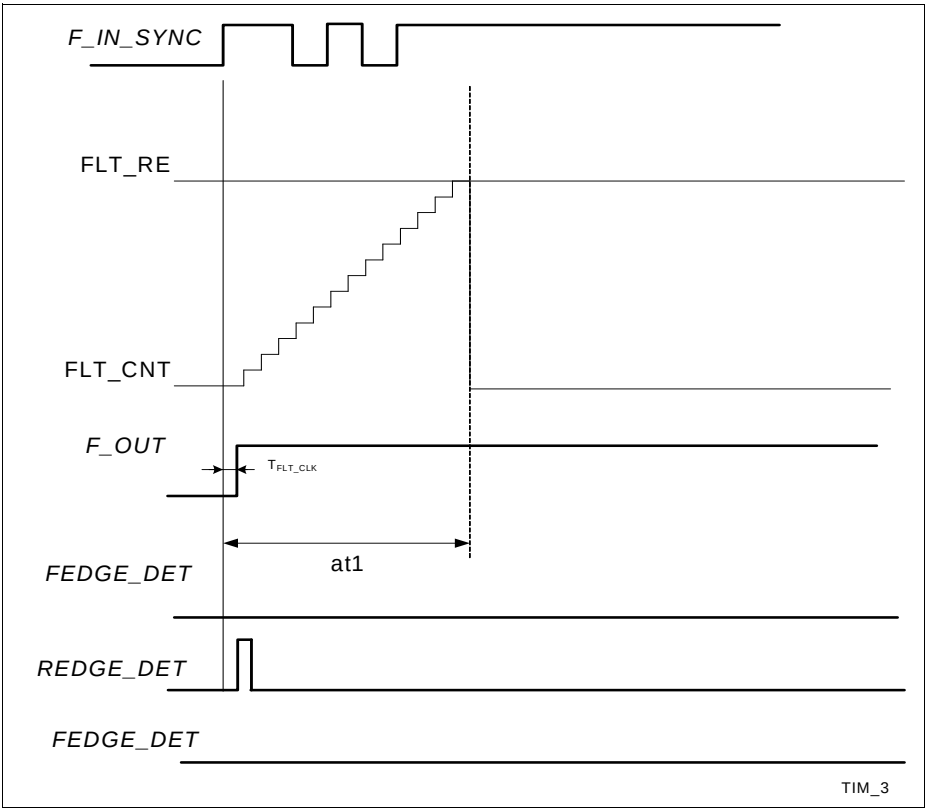


Figure 24-21 Immediate Edge Propagation Mode in the case of a rising edge

In immediate edge propagation mode the glitch measurement mechanism is not applied to the edge detection. Detected edges on F_IN_SYNC are transferred directly to F_OUT.

The counter FLT_CNT is increment until acceptance time threshold is reached.

Figure 24-22 shows a more complex example of the TIM filter, in which both, rising and falling edges are configured in immediate edge propagation mode.

Immediate Edge Propagation Mode in the case of a rising and falling edge

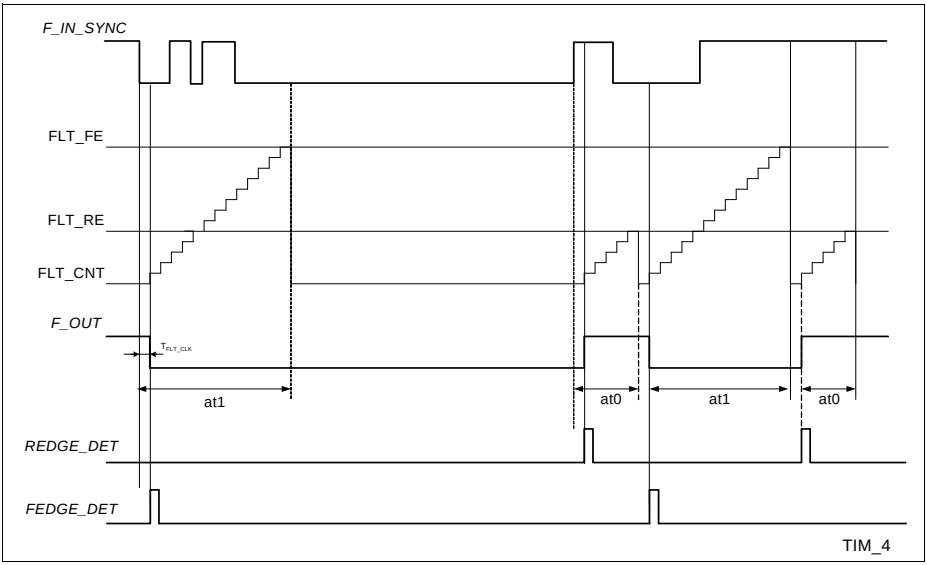


Figure 24-22 Immediate Edge Propagation Mode in the case of a rising and falling edge

If the FLT_CNT has reached the acceptance time for a specific signal edge and the signal F_IN_SYNC has already changed to the opposite level of F_OUT, the opposite signal level is set to F_OUT and the acceptance time measurement is started immediately. **Figure 24-22** shows this scenario at the detection of the first rising edge and the second falling edge.

Individual De-Glitch Time Mode (up/down counter)

In individual de-glitch time mode (up/down counter) each edge of an input signal can be filtered with an individual de-glitch threshold filter value mentioned in the registers FLT_RE and FLT_FE, respectively.

The filter counter register FLT_CNT is increment when the signal level on F_IN_SYNC is unequal to the signal level on F_OUT and decremented if F_IN_SYNC equals F_OUT.

If After FLT_CNT has reached a value of zero during decrease the counter is stopped immediately.

If a glitch is detected a glitch detection bit GLITCHDET is set in the TIM0_CHx_IRQ_NOTIFY register.

The detected edge signal together with the new signal level is propagated to F_OUT after the individual de-glitch threshold is reached. **Figure 24-23** shows the behaviour of the filter in individual de-glitch time (up/down counter) mode in the case of the rising edge detection.

Individual Deglitch Time Mode (up/down counter) in the case of a rising edge

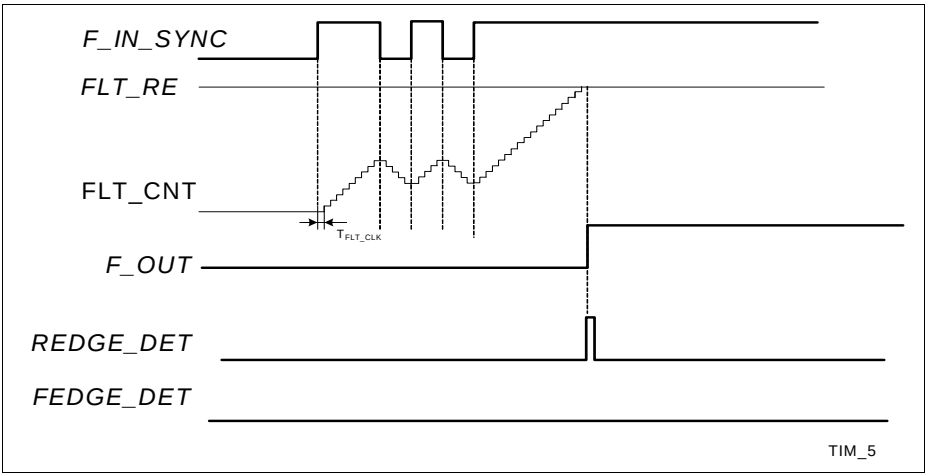


Figure 24-23 Individual Deglitch Time Mode (up/down counter) in the case of a rising edge

Individual De-Glitch Time Mode (hold counter)

In individual de-glitch time mode (hold counter) each edge of an input signal can be filtered with an individual de-glitch threshold filter value mentioned in the registers FLT_RE and FLT_FE, respectively.

The filter counter register FLT_CNT is increment when the signal level on F_IN_SYNC is unequal to the signal level on F_OUT and the counter value of FLT_CNT is hold if F_IN equals F_OUT.

If a glitch is detected the glitch detection bit GLITCHDET is set in the TIM0_CHx_IRQ_NOTIFY register.

The detected edge signal together with the new signal level is propagated to F_OUT after the individual de-glitch threshold is reached. **Figure 24-24** shows the behaviour of the filter in individual de-glitch time (hold counter) mode in the case of the rising edge detection.

Individual Deglitch Time Mode (hold counter) in the case of a rising edge

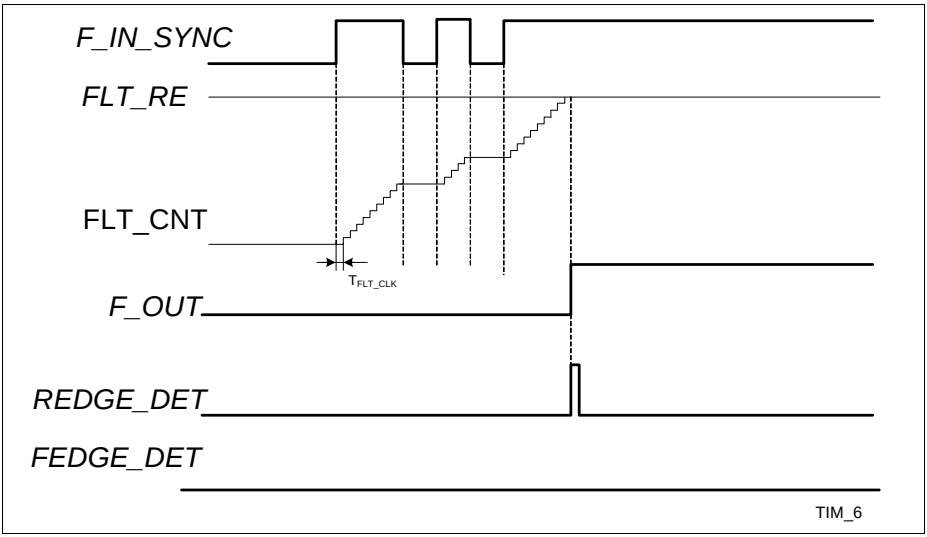


Figure 24-24 Individual Deglitch Time Mode (hold counter) in the case of a rising edge

Immediate Edge Propagation and Individual De-Glitch Mode

As already mentioned, the three different filter modes can be applied individually to each edge of the measured signal.

However, if one edge is configured with immediate edge propagation and the other edge with an individual deglitch mode (whether up/down counter or hold counter) a special consideration has to be applied.

Assume that the rising edge is configured for immediate edge propagation and the falling edge with individual deglitch mode (up/down counter) as shown in **Figure 24-25**.

If the falling edge of the incoming signal already occurs during the measuring of the acceptance time of the rising edge, the measurement of the deglitch time on the falling edge is started delayed, but immediately after the acceptance time measurement phase of the rising edge has finished.

Consequently, the deglitch counter can not measure the time TERROR, as shown in **Figure 24-25**.

Mixed mode measurement

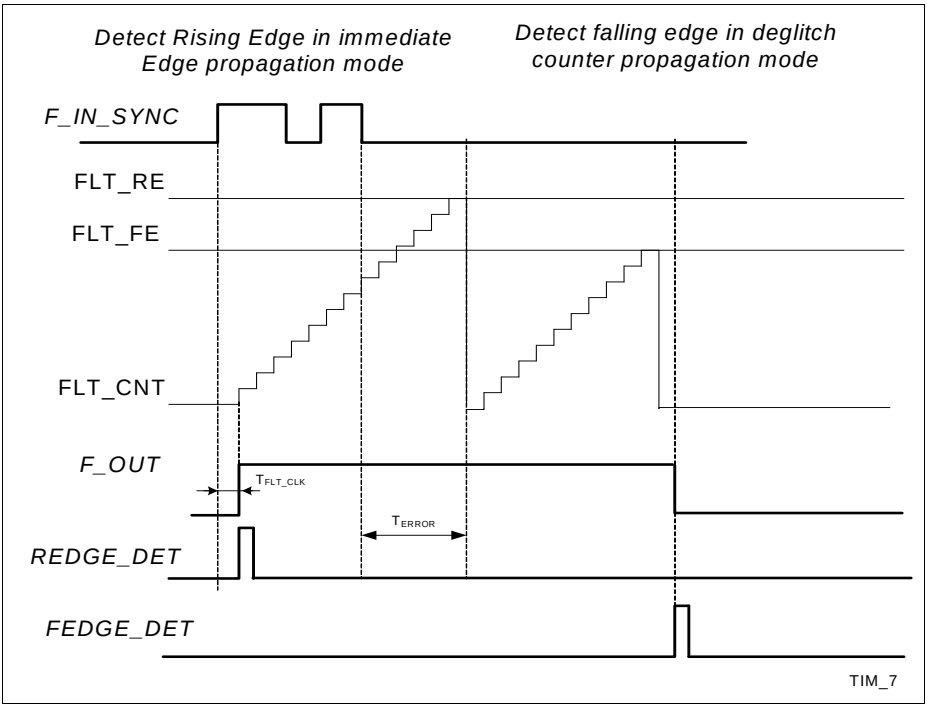


Figure 24-25 Mixed mode measurement

24.5.2.3 TIM Filter reconfiguration

If FLT_EN=1 a change of FLT_RE or FLT_FE will take place immediately. If FLT_EN=1 a change of FLT_MODE_RE or FLT_FE will be used with the next occurring corresponding edge. If mode is changed while the filter unit is processing a certain mode, it will end this edge filtering in the mode as started.

If FLT_EN=1 a change of FLT_DTR_RE or FLT_CTR_FE will take place immediately.

24.5.3 Timeout Detection Unit (TDU)

The Timeout Detection Unit (TDU) is responsible for timeout detection of the TIM input signals.

Each channel of the TIM submodule has its own Timeout Detection Unit (TDU) where a timeout event can be set up on the filtered input signal of the corresponding channel.

The TDU architecture is shown in **Figure 24-26**.

24.5.3.1 Architecture of the TDU Subunit

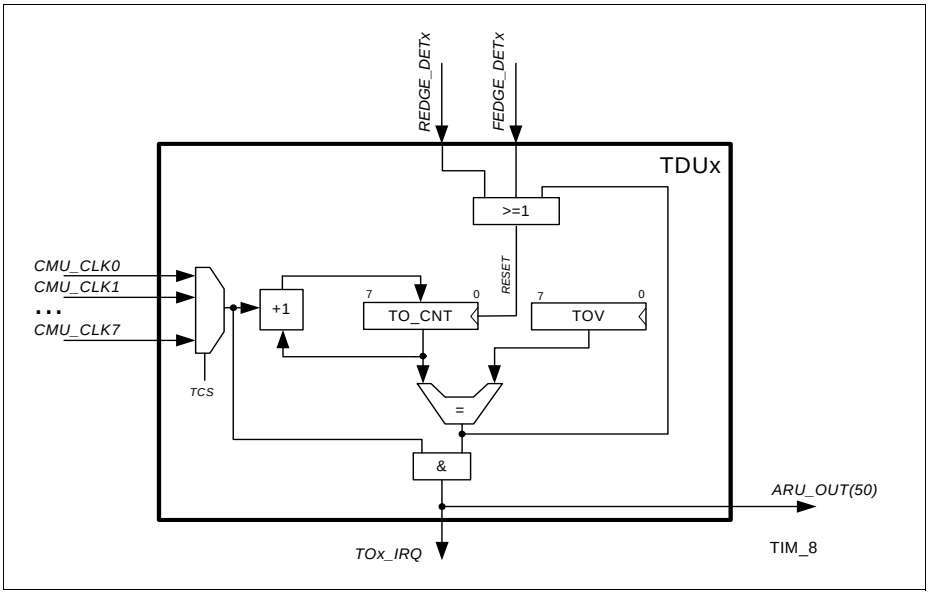


Figure 24-26 Architecture of the TDU Subunit

It is possible to detect timeouts with the resolution of the specified CMU_CLKx input signal selected with the bit field TCS of the register TIM0_CHx_TDU. The individual timeout values have to be specified in number of ticks of the selected input clock signal and have to be specified in the field TOV of timeout value register TIM0_CHx_TDUV of the TIM channel x (x:0…7).

The exact time out value TTDU can be calculated with:

$$T_{TDU}=(TOV+1)*T_{CMU_GCLKx},$$

whereas T_{CMU_GCLKx} is the clock period of the selected CMU clock signal.

Timeout detection can be enabled or disabled individually inside the TIM0_CHx_CTRL register by setting/resetting the TO_CTRL bit.

Timeout detection can be enabled to be sensitive to falling, rising or both edges of the input signal by writing the corresponding values to the bit field TOCTRL.

The counter TO_CNT is reset by each detected input edge coming either from the filtered input signal or when the timeout value TOV is reached by the counter TO_CNT.

After such a reset or by enabling the channel inside the TIM0_CHx_CTRL register the counter TO_CNT starts counting again with the specified clock input signal.

Otherwise, timeout measurements starts immediately after the TO_CTRL bit inside the TIM0_CHx_CTRL register is written (enabled).

The TDU generates an interrupt signal TIM_TODETx_IRQ whenever a timeout is detected for an individual input signal, and the TODET bit is set inside the TIM0_CHx_IRQ_NOTIFY register.

24.5.4 TIM Channel Architecture

24.5.4.1 Overview

Each TIM channel consist of an input edge counter ECNT, a Signal Measurement Unit (SMU) with a counter CNT, a counter shadow register CNTS for SMU counter and two general purpose registers GPR0 and GPR1 for value storage.

The value TOV of the timeout register TIM0_CHx_TDU is provided to TDU subunit of each individual channel for timeout measurement. The architecture of the TIM channel is depicted in **Figure 24-27**.

TIM Channel Architecture

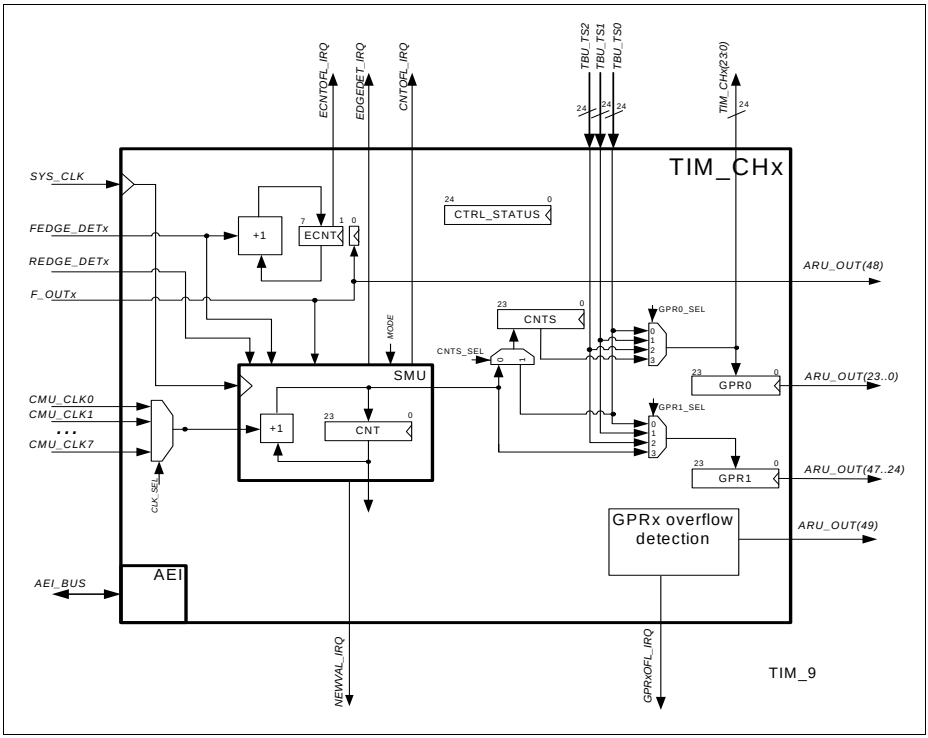


Figure 24-27 TIM Channel Architecture

Each TIM channel receives both input trigger signals REDGE_DETx and FEDGE_DETx, generated by the corresponding filter module in order to signalize a detected echo of the input signal F_INx. The signal F_OUTx shows the filtered signal of the channel's input signal F_INx.

The edge counter ECNT counts every incoming filtered edge (rising and falling). The counter value is uneven in case of detected rising, and even in case of detected falling edge. Thus, the input signal level is part of the counter and can be obtained by bit 0 of ECNT. (However, the actual counter implementation counts only falling edges on ECNT[n:1] bits. It generates ECNT by composing the ECNT[n:1] bits with F_OUTx as bit 0).

Thus, the whole counter value is always odd, when a positive edge was received and always even, when a negative edge was received.

The current ECNT[7:0] register content is made visible on the bits 31 down to 24 of the GPRz and CNTS registers. This allows the software to detect inconsistent read accesses to registers GPR0, GPR1, and CNTS. However, the update strategy of these

registers depends on the selected TIM modes, and thus the consistency check has to be adapted carefully.

It can be chosen with the bit field FR_ECNT_OFL when an ECNT overflow is signaled on ECNTOFL. An ECNT overflow can be signaled on 8 bit or full range resolution.

While reading the register TIM0_CHx_ECNT the bit ECNT[0] shows the input signal value F_OUTx independent of the state (enabled / disabled) of the channel. If a channel gets disabled (OSM mode or resetting TIM_EN) the content of TIM0_CHx_ECNT will be frozen until a read of the register takes place. this read will reset the ECNT counter. Continuing reads will show the input signal in bit ECNT[0] again.

When new data is written into GPR0 and GPR1 the NEWVAL bit is set in TIM0_CHx_IRQ_NOTIFY register and depending on corresponding enable bit value the NEWVALx_IRQ interrupt is raised.

To guarantee a consistent delivery of data from the GPR0 and GPR1 registers to the CPU each TIM channel has to ensure that the data is consumed before it is overwritten with new values.

If new data was produced by the TIM channel (bit NEWVAL is set inside TIM0_CHx_IRQ_NOTIFY register) while the old data is not consumed by the CPU, the TIM channel sets the GPRzOFL bit inside the status register TIM0_CHx_IRQ_NOTIFY and it overwrites the data inside the GPRz registers.

With the CPU as consumer for the GPRz registers, the acknowledge for reading out data is performed by a read access to the register GPR0. Thus, register GPR1 should be read always before GPR0.

24.5.4.2 TIM Channel Modes

The TIM provides six different measurement modes that can be configured with the bit field TIM_MODE of register TIM0_CHx_CTRL. The measurement modes are described in the following subsections. Besides these different basic measurement modes, there exist distinct configuration bits in the register TIM0_CHx_CTRL for a more detailed controlling of each mode. The meanings of these bits are as follows:

- DSL: control the signal level for the measurement modes (e.g. if a measurement is started with rising edge or falling edge, or if high level pulses or low level pulses are measured.
- EGPR0_SEL, GPR0_SEL and EGPR1_SEL, GPR1_SEL: control the actual content of the registers GPR0 and GPR1 after a measurement has finished.
- CNTS_SEL: control the content of the registers CNTS. The actual time for updating the CNTS register is mode dependent.
- OSM: activate measurement in one-shot mode or continuous mode. In one-shot mode only one measurement cycle is performed and after that the channel is disabled.

- NEWVAL: The NEWVAL IRQ interrupt is triggered at the end of a measurement cycle, signalling that the registers GPR0 and GPR1 are updated.
- EXT_CAP_EN: forces an update of the registers GPR0 and GPR1 and CNTS (TIM channel mode dependant) only on each rising edge of the EXT_CAPTURE signal and triggers a NEWVAL IRQ interrupt. If this mode is disabled the NEWVAL IRQ interrupt is triggered at the end of each measurement cycle.

For each channel the source of the EXT_CAPTURE signal can be configured with the bit fields EXT_CAP_SRC in the register TIM0_CHx_ECTRL.

TIM PWM Measurement Mode (TPWM)

In TIM PWM Measurement Mode the TIM channel measures duty cycle and period of an incoming PWM signal. The DSL bit defines the polarity of the PWM signal to be measured.

When measurement of pulse high time and period is requested (PWM with a high level duty cycle, DSL=1), the channel starts measuring after the first rising edge is detected by the filter.

Measurement is done with the CNT register counting with the configured clock coming from CMU_CLKx until a falling edge is detected.

Then the counter value is stored inside the shadow register CNTS (if CNTS_SEL = 0) and the counter CNT counts continuously until the next rising edge is reached.

On this following rising edge the content of the CNTS register is transferred to GPR0 and the content of CNT register is transferred to GPR1, assuming settings for the selectors GPR0_SEL=11 and GPR1_SEL=11. By this, GPR0 contains the duty cycle length and GPR1 contains the period. It should be noted, that the bits 1 to 7 of the ECNT may be used to check data consistency of the registers GPR0 and GPR1.

In addition the CNT register is cleared NEWVAL status bit inside of TIM0_CHx_IRQ_NOTIFY status register and depending on corresponding interrupt enable condition TIM_NEWVALx_IRQ interrupt is raised.

If a PWM with a low level duty cycle should be measured (DSL = 0), the channel waits for a falling edge until measurement is started. On this edge the low level duty cycle time is stored first in CNTS and then finally in GPR0 and the period is stored in GPR1.

When a PWM period was successfully measured, the data in GPRz registers is marked as valid for reading, the NEWVAL bit is set inside the TIM0_CHx_IRQ_NOTIFY register, and a new measurement is started.

If the preceding PWM values were not consumed by a reader (CPU) the TIM channel set GPRzOFL status bit in TIM0_CHx_IRQ_NOTIFY and depending on corresponding interrupt enable bit value raises a GPRzOFLx_IRQ and overwrites the old values in GPR0 and GPR1. A new measurement is started afterwards.

If the register CNT produces an overflow during the measurement, the bit CNTOFL is set inside the register TIM0_CHx_IRQ_NOTIFY and interrupt TIM_CNTOFL[x]_IRQ is raised depending on corresponding interrupt enable condition.

If the register ECNT produces an overflow during the measurement, the bit ECNTOFL is set inside the register TIM0_CHx_IRQ_NOTIFY and interrupt TIM_ECNTOFL[x]_IRQ is raised depending on corresponding interrupt enable condition.

External capture TIM PWM Measurement Mode (TPWM)

If external capture is enabled, the pwm measurement is done continuously. The actual measurement values are captured to GPRx if an external capture event occurs.

Operation is done depending on cmu clock, ISL, DSL bit and the input signal value defined in the next table:

Table 24-10 TIM PWM Measurement Mode

Input signal F_OUTx	selected CMU Clock	External capture	ISL	DSL	Action description
0	1	0	-	0	CNT++
1	1	0	-	0	no
rising edge	-	0	0	0	capture CNT value in CNTS
falling edge	-	0	0	0	CNT = 0
rising edge	-	0	1	0	no
falling edge	-	0	1	0	capture CNT value in CNTS; CNT = 0
1	1	0	-	1	CNT++
0	1	0	-	1	no
falling edge	-	0	0	1	capture CNT value in CNTS
rising edge	-	0	0	1	CNT = 0
falling edge	-	0	1	1	no
rising edge	-	0	1	1	capture CNT value in CNTS; CNT = 0
-	-	rising edge	-	-	do GPRx capture; issue NEWVAL_IRQ
-	0	0	-	-	no

TIM Pulse Integration Mode (TPIM)

In TIM Pulse Integration Mode each TIM channel is able to measure a sum of pulse high or low times on an input signal, depending on the selected signal level bit DSL of register TIM0_CHx_CTRL register.

The pulse times are measured by incrementing the TIM channel counter CNT whenever the pulse has the specified signal level DSL. The counter is stopped whenever the input signal has the opposite signal level.

The counter CNT counts with the CMU_CLKx clock specified by the CLK_SEL bit field of the TIM0_CHx_CTRL register.

The CNT register is reset at the time the channel is activated (enabling via AEI write access) and it accumulates pulses while the channel is staying enabled.

Whenever the counter is stopped, the registers CNTS, GPR0 and GPR1 are updated according to settings of its corresponding input multiplexers, using the bits GPR0_SEL, EGPR0_SEL, EGPR1_SEL, GPR1_SEL, and CNTS_SEL. It should be noted, that the bits 1 to 7 of the ECNT may be used to check data consistency of the registers GPR0 and GPR1.

External capture TIM Pulse Integration Mode (TPIM)

If external capture is enabled, the pulse integration is done until next external capture event occurs.

Operation is done depending on CMU clock, DSL bit and the input signal value defined in the next table:

Table 24-11 TIM integration Mode

Input signal F_OUTx	selected CMU Clock	External capture	ISL	DSL	Action description
0	1	0	-	0	CNT++
1	1	0	-	0	no
1	1	0	-	1	CNT++
0	1	0	-	1	no
-	-	rising edge	-	-	do GPRx capture; issue NEWVAL_IRQ
-	0	0	-	-	no

TIM Input Event Mode (TIEM)

In TIM Input Event Mode the TIM channel is able to count edges.

It is configurable if rising, falling or both edges should be counted. This can be done with the bit fields DSL and ISL in TIM0_CHx_CTRL register.

In addition, a TIM0_NEWVAL[x]_IRQ interrupt is raised when the configured edge was received and this interrupt was enabled.

The counter register CNT is used to count the number of edges, and the bit fields EGPR0_SEL, EGPR1_SEL, GPR0_SEL, GPR1_SEL, and CNTS_SEL can be used to configure the desired update values for the registers GPR0, GPR1 and CNTS. These register are updated whenever the edge counter CNT is increment due to the arrival of a desired edge.

GPRz If the preceding data was not consumed by the CPU the TIM channel sets GPRzOFL status bit and raises a GPRzOFL[x]_IRQ if it was enabled in TIM0_CHx_IRQ_EN register and overwrites the old values in GPR0 and GPR1 with the new ones.

If the register CNT produces an overflow during the measurement, the bit CNTOFL is set inside the register TIM0_CHx_IRQ_NOTIFY and interrupt TIM_CNTOFL[x]_IRQ is raised depending on corresponding interrupt enable condition.

If the register ECNT produces an overflow during the measurement, the bit ECNTOFL is set inside the register TIM0_CHx_IRQ_NOTIFY and interrupt TIM_ECNTOFL[x]_IRQ is raised depending on corresponding interrupt enable condition.

The TIM Input Event Mode does not depend on the bit field CLK_SEL of register TIM0_CHx_CTRL.

External capture TIM Input Event Mode (TIEM)

If external capture is enabled, capturing is done depending on ISL, DSL bit and the input signal value defined in the next table:

Table 24-12 TIM Input Event Mode

Input signal F_OUTx	External capture	ISL	DSL	Action description
-	rising edge	1	-	do capture; issue NEWVAL_IRQ; CNT++
-	0	-	-	no
1	rising edge	0	1	do capture; issue NEWVAL_IRQ; CNT++
0	-	0	1	no
0	rising edge	0	0	do capture; issue NEWVAL_IRQ; CNT++
1	-	0	0	no

TIM Input Prescaler Mode (TIPM)

In the TIM Input Prescaler Mode the number of edges which should be detected before a TIM0_NEWVAL[x]_IRQ is raised is programmable. In this mode it must be specified in the CNTS register after how many edges the interrupt has to be raised.

A value of 0 in CNTS means that after one edge an interrupt is raised, and a value of 1 means that after two edges an interrupt is raised, and so on.

The edges to be counted can be selected by the bit fields DSL and ISL of register TIM0_CHx_CTRL.

With each triggered interrupt, the registers GPR0 and GPR1 are updated according to bits EGPR0_SEL, EGPR1_SEL, GPR0_SEL and GPR1_SEL.

If the register ECNT produces an overflow during the measurement, the bit ECNTOFL is set inside the register TIM0_CHx_IRQ_NOTIFY and interrupt TIM_ECNTOFL[x]_IRQ is raised depending on corresponding interrupt enable condition.

The TIM Input Prescaler Mode does not depend on the bit field CLK_SEL of register TIM0_CHx_CTRL.

External capture TIM Input Prescaler Mode (TIPM)

If external capture is enabled, the external capture events are counted instead of the input signal edges.

Operation is done depending on DSL, ISL bit and the input signal value defined in the next table:

Table 24-13 TIM Input Event Mode

Input signal F_OUTx	External capture	ISL	DSL	Action description
-	rising edge	1	-	if CNT == CNTS then do capture; issue NEWVAL_IRQ; CNT = 0; else CNT++ endif
-	0	1	-	no
1	rising edge	0	1	if CNT == CNTS then do capture; issue NEWVAL_IRQ; CNT = 0; else CNT++ endif
0	-	0	1	no
0	rising edge	0	0	if CNT == CNTS then do capture; issue NEWVAL_IRQ; CNT = 0; else CNT++ endif
1	-	0	0	no

TIM Gated Periodic Sampling Mode (TGPS)

In the TIM Gated Periodic Sampling Mode the number of CMU clock cycles which should elapse before capturing and rising TIM0_NEWVALx_IRQ is programmable. In this mode it must be specified in the CNTS register after how many CMU clock cycles the interrupt has to be raised.

A value of 0 in TIM0_CHx_CNTS means that after one CLK_SEL edge a trigger/interrupt is raised, and a value of 1 means that after two edges a trigger/interrupt is raised, and so on.

In the TIM0_CHx_CNT register the elapsed cycles were incremented and compared against TIM0_CHx_CNTS. If TIM0_CHx_CNT is greater or equal to TIM0_CHx_CNTS a trigger will be raised. This allows by writing a value to TIM0_CHx_CNTS that the actual period time can be changed on the fly.

Operation is done depending on CMU clock, DSL, ISL bit and the input signal value defined in the next table:

Table 24-14 TIM Gated Periodic Sampling Mode

Input signal F_OUTx	selected CMU Clock	External capture	ISL	DSL	Action description
-	1	0	1	-	if CNT == CNTS then do capture; issue NEWVAL_IRQ; CNT = 0; else CNT++ endif
0	0	0	0	1	no
1	1	0	0	1	if CNT == CNTS then do capture; issue NEWVAL_IRQ; CNT = 0; else CNT++ endif
0	0	-	0	1	no
0	1	0	0	0	if CNT == CNTS then do capture; issue NEWVAL_IRQ; CNT = 0; else CNT++ endif
1	0	0	0	0	no
-	0	0	-	-	no

In this mode the TIM0_CHx_GPR1 operates as a shadow register for TIM0_CHx_CNTS. This would allow that the period for the next sampling period could be specified. The update of TIM0_CHx_CNTS will only take place once on a trigger if the TIM0_CHx_GPR1 was written by the CPU. This means that the capture value from the

previous trigger can be read by the CPU from TIM0_CHx_GPR1 and afterwards the new sampling period (the one after the actual sampling period) could be written.

With each triggered interrupt, the registers GPR0 and GPR1 are updated according to bits GPR0_SEL, GPR1_SEL, EGPR0_SEL and EGPR1_SEL.

When selecting ECNT as a source for the capture registers, GPRx will show the edge count and the input signal value at point of capture. Selecting GPR0_SEL = 11_B and EGPR0_SEL = 0_B for TIM channel 0 all 8 TIM input signals will captured to GPR[7:0].

In the TGPS Mode the bit field CLK_SEL of register TIM0_CHx_CTRL will define the selected CMU clock which will be used.

The behaviour of the ECNT counter is configurable by ECNT_RESET. If set to 1 on each interrupt (period expired) the ECNT will be reset. Otherwise it operates in wrap around mode.

If the register ECNT produces an overflow during the measurement, the bit ECNTOFL is set inside the register TIM0_CHx_IRQ_NOTIFY and interrupt TIM_ECNTOFLx_IRQ is raised depending on corresponding interrupt enable condition.

External capture Bit Gated Periodic Sampling Mode (TGPS)

If external capture is enabled, the external capture events will capture the GPRx, reset the counter CNT and issue a NEWVAL_IRQ.

Operation is done depending on CMU clock, DSL, ISL bit and the input signal value defined in the next table:

Table 24-15 TIM Gated Periodic Sampling Mode

Input signal F_OUTx	selected CMU Clock	External capture	ISL	DSL	Action description
-	1	0	1	-	if CNT == CNTS then do capture; issue NEWVAL_IRQ; CNT = 0; else CNT++ endif
0	0	0	0	1	no
1	1	0	0	1	if CNT == CNTS then do capture; issue NEWVAL_IRQ; CNT = 0; else CNT++ endif
0	0	-	0	1	no

Table 24-15 TIM Gated Periodic Sampling Mode (cont'd)

Input signal F_OUTx	selected CMU Clock	External capture	ISL	DSL	Action description
0	1	0	0	0	if CNT == CNTS then do capture; issue NEWVAL_IRQ; CNT = 0; else CNT++ endif
1	0	0	0	0	no
-	0	0	-	-	no
-	-	rising edge	-	-	do capture; issue NEWVAL_IRQ; CNT = 0

24.5.5 TIM Interrupt Signals

TIM provides 6 interrupt lines per channel. These interrupts are shown below:

Table 24-16 TIM Interrupt Signals

Signal	Description
TIM0_NEWVAL[x]_IRQ	New measurement value detected by SMU of channel x (x:0…7)
TIM0_ECNTOFL[x]_IRQ	ECNT counter overflow of channel x (x:0…7)
TIM0_CNTOFL[x]_IRQ	SMU CNT counter overflow of channel x (x:0…7)
TIM0_GPRzOF[x]_IRQ	GPRz data overflow, old data was not read out before new data has arrived at input pin (x:0…7)
TIM0_TODET[x]_IRQ	Time out reached for input signal of channel x (x:0…7)
TIM0_GLITCHDET_IRQ	A glitch was detected by the TIM filter of channel (x: 0…7).

24.5.6 TIM Configuration Registers Overview

TIM contains following configuration registers:

Table 24-17 TIM Configuration Registers Overview

Register Name	Description	Detail in Section
TIM0_CHx_CTRL	TIM channel x control register (x:0…7)	Section 24.5.7.1
TIM0_CHx_ECTRL	TIM channel x (x:0…7) extended control	Section 24.5.7.18
TIM0_CHx_FLT_RE	TIM channel x filter parameter 0 register (x:0…7)	Section 24.5.7.2
TIM0_CHx_FLT_FE	TIM channel x filter parameter 1 register (x:0…7)	Section 24.5.7.3
TIM0_CHx_TDUV	TIM channel x TDU control register (x:0…7)	Section 24.5.7.15
TIM0_CHx_TDUC	TIM channel x TDU counter register (x:0…7)	Section 24.5.7.16
TIM0_CHx_GPR0	TIM channel x general purpose 0 register (x:0…7)	Section 24.5.7.4
TIM0_CHx_GPR1	TIM channel x general purpose 1 register (x:0…7)	Section 24.5.7.5
TIM0_CHx_CNT	TIM channel x SMU counter register (x:0…7)	Section 24.5.7.6
TIM0_CHx_ECNT	TIM channel x (x:0…7) SMU edge counter	Section 24.5.7.17
TIM0_CHx_CNTS	TIM channel x SMU shadow counter register (x:0…7)	Section 24.5.7.7
TIM0_CHx_IRQ_NOTIFY	TIM channel x interrupt notification register (x:0…7)	Section 24.5.7.8
TIM0_CHx_IRQ_EN	TIM channel x interrupt enable register (x:0…7)	Section 24.5.7.9
TIM0_CHx_EIRQ_EN	TIM channel x (x:0…7) error interrupt enable	Section 24.5.7.14
TIM0_CHx_IRQ_FORCINT	TIM channel x software interrupt force register (x:0…7)	Section 24.5.7.10
TIM0_CHx_IRQ_MODE	TIM IRQ mode configuration register (x=0…7)	Section 24.5.7.11

Table 24-17 TIM Configuration Registers Overview (cont'd)

Register Name	Description	Detail in Section
TIM0_RST	TIM global software reset register	Section 24.5.7.12
TIM0_IN_SRC	TIM AUX IN source selection	Section 24.5.7.13
TIM0_IN_VAL	TIM input value observation	Section 24.5.7.19

24.5.7 TIM Configuration Registers Description

All of the following registers are 32-bit only accessible.

24.5.7.1 Register TIM0_CHx_CTRL (x:0…7)

GTM_TIM0_CHx_CTRL (x=0-7)
TIM Channel x Control Register (01024_H+x*80_H) **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TOCTRL		EGPR1_SEL	EGPR0_SEL	FR_ECNT_OFL	CLK_SEL			FLT_CTR_FE	FLT_MODE_FE	FLT_CTR_RE	FLT_MODE_RE	Reserved	FLT_CNT_FRQ		FLT_EN
rw		rw	rw	rw	rw			rw	rw	rw	rw	r	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECNT_RESET	ISL	DSL	CNTS_SEL	GPR1_SEL		GPR0_SEL		TBU0x_SEL	CICTRL	0	OSM	TIM_MODE			TIM_EN
rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw			rwh

Field	Bits	Type	Description
TIM_EN	0	rwh	**TIM channel x (x:0…7)enable** 0_B Channel disabled 1_B Channel enabled Note: Enabling of the channel resets the registers ECNT, TIM0_CHx_CNT, TIM0_CHx_GPR0, and TIM0_CHx_GPR1 to their reset values. Note: After finishing the action in one-shot mode the TIM_EN bit is cleared automatically. Otherwise, the bit must be cleared manually.

Field	Bits	Type	Description
TIM_MODE	[3:1]	rw	**TIM channel x (x:0…7) mode** 000_B PWM Measurement Mode (TPWM) 001_B Pulse Integration Mode (TPIM) 010_B Input Event Mode (TIEM) 011_B Input Prescaler Mode (TIPM) 100_B Reserved 101_B Gated Periodic Sampling Mode (TGPS) Note: If an undefined value is written to the TIM_MODE register, the hardware switches automatically to TIM_MODE = 000 (TPWM mode). Note: The TIM_MODE register should not be changed while the TIM channel is enabled. Note: If TIM channel is enabled and operating in TPWM or TPIM mode after the first valid edge defined by DSL has occurred, a reconfiguration of DSL, ISL, and TIM_MODE will not change the channel behavior. Reading these bit fields after reconfiguration will show the newly configured settings but the initial channel behavior will not change. Only a disabling of the TIM_EN=0 will change the channel operation mode.
OSM	4	rw	**One-shot mode** 0_B Continuous operation mode 1_B One-shot mode Note: After finishing the action in one-shot mode the TIM_EN bit is cleared automatically.
0	5	rw	**Reserved** Have to be written as zero
CICTRL	6	rw	**Channel Input Control** 0_B use signal TIM_IN(x) as input for channel x 1_B use signal TIM_IN(x-1) as input for channel x (or TIM_IN(7) if x is 0)
TBU0x_SEL	7	rw	**TBU_TS0 bits input select for TIM_CHx_GPRz (x: 0, 1)** 0_B Use TBU_TS0(23…0) to store in TIM0_CHx_GPRz 1_B Use TBU_TS0(26…3) to store in TIM0_CHx_GPRz Note: This bit is only applicable for TIM0

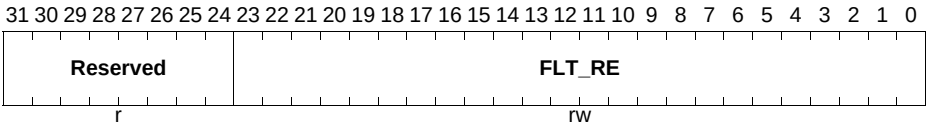
Field	Bits	Type	Description
GPR0_SEL	[9:8]	rw	**Selection for GPR0 register** If EGPR0_SEL =0: 00_B use TBU_TS0 as input 01_B use TBU_TS1 as input 10_B use TBU_TS2 as input 11_B use CNTS as input; in TGPS mode in channel = 0 use TIM filter F_OUT as input If EGPR0_SEL =1: 00_B use ECNT as input 01_B use TIM_INP_VAL as input 10_B reserved 11_B reserved *Note: In TBCM mode: EGPR1_SEL=1, GPR1_SEL=01 selects TIM_INP_VAL as input, in all cases TIM filter F_OUT is used.*
GPR1_SEL	[11:10]	rw	**Selection for GPR1 register** If EGPR0_SEL =0: 00_B use TBU_TS0 as input 01_B use TBU_TS1 as input 10_B use TBU_TS2 as input 11_B use CNT as input If EGPR0_SEL =1: 00_B use ECNT as input 01_B use TIM_INP_VAL as input 10_B reserved 11_B reserved *Note: In TBCM mode EGPR1_SEL, GPR1_SEL are ignored TIM Filter F_OUT is used as input.* *Note: If a reserved value is written to the EGPR1_SEL, GPR1_SEL bit fields, the hardware will use TBU_TS0 input.*
CNTS_SEL	12	rw	**Selection for CNTS register** 0_B use CNT register as input 1_B use TBU_TS0 as input Note: The functionality of the CNTS_SEL is disabled in the modes TIPM and TBCM.

Field	Bits	Type	Description
DSL	13	rw	**Signal level control** 0_B Measurement starts with falling edge (low level measurement) 1_B Measurement starts with rising edge (high level measurement)
ISL	14	rw	**Ignore signal level** 0_B use DSL bit for selecting active signal level 1_B ignore DSL and treat both edges as active edge Note: This bit is only applicable in Input Event mode (TIEM and TIPM)
ECNT_RESET	15	rw	**Enables resetting the ECNT counter in periodic sampling mode** 0_B ECNT counter operating in wrap around mode 1_B ECNT counter is reset with periodic sampling
FLT_EN	16	rw	**Filter enable for channel x (x:0…7)** 0_B Filter disabled and internal states are reset 1_B Filter enabled Note: If the filter is disabled all filter related units (including CSU) are bypassed, which means that the signal F_IN is directly routed to signal F_OUT.
FLT_CNT_FRQ	[18:17]	rw	**Filter counter frequency select** 00_B FLT_CNT counts with CMU_CLK0 01_B FLT_CNT counts with CMU_CLK1 10_B FLT_CNT counts with CMU_CLK6 11_B FLT_CNT counts with CMU_CLK7
EXT_CAP_EN	19	rw	**Enables external capture mode** The selected TIM mode is only sensitive to external capture pulses the input event changes are ignored 0_B External capture disabled 1_B External capture enabled
FLT_MODE_RE	20	rw	**Filter mode for rising edge** 0_B Immediate edge propagation mode 1_B individual de-glitch mode
FLT_CTR_RE	21	rw	**Filter counter mode for rising edge** 0_B Up/Down Counter 1_B Hold Counter Note: This bit is only applicable in Individual Deglitch Time Mode

Field	Bits	Type	Description
FLT_MODE_FE	22	rw	**Filter mode for falling edge** 0_B Immediate edge propagation mode 1_B individual de-glitch mode
FLT_CTR_FE	23	rw	**Filter counter mode for falling edge** 0_B Up/Down Counter 1_B Hold Counter Note: This bit is only applicable in Individual Deglitch Time Mode
CLK_SEL	[26:24]	rw	**CMU clock source select for channel** 000_B CMU_CLK0 selected 001_B CMU_CLK1 selected 010_B CMU_CLK2 selected 011_B CMU_CLK3 selected 100_B CMU_CLK4 selected 101_B CMU_CLK5 selected 110_B CMU_CLK6 selected 111_B CMU_CLK7 selected
FR_ECNT_OFL	27	rw	**Extended Edge counter overflow behaviour** 0_B Overflow will be signalled on ECNT bit width = 8 1_B Overflow will be signalled on EECNT bit width (full range)
EGPR0_SEL	28	rw	**Extension of GPR0_SEL bit field** Details described in GPR0_SEL bit field.
EGPR1_SEL	29	rw	**Extension of GPR1_SEL bit field** Details described in GPR1_SEL bit field.
TOCTRL	[31:30]	rw	**Timeout control** 00_B Timeout feature disabled 01_B Timeout feature enabled for both edges 10_B Timeout feature enabled for rising edge only 11_B Timeout feature enabled for falling edge only

24.5.7.2 Register TIM0_CHx_FLT_RE(x:0…7)

GTM_TIM0_CHx_FLT_RE (x=0-7)
GTM_TIM0 Channel x Filter Parameter 0 Register
($0101C_H$+x*80_H) **Reset Value: 00000000_H**

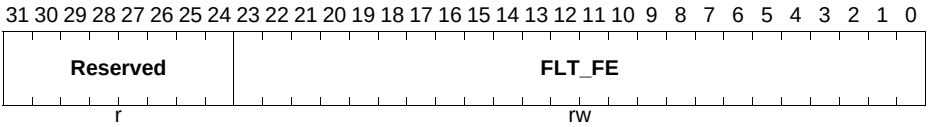


Field	Bits	Type	Description
FLT_RE	[23:0]	rw	**Filter parameter for rising edge** Note: This register has different meanings in the various filter modes. Immediate edge propagation mode = acceptance time for rising edge Individual deglitch time mode = deglitch time for rising edge
Reserved	[31:24]	r	**Reserved** Read as zero, should be written as zero

24.5.7.3 Register TIM0_CHx_FLT_FE (x:0…7)

GTM_TIM0_CHx_FLT_FE (x=0-7)
TIM0 Channel x Filter Parameter 1 Register

(01020_H+x*80_H) **Reset Value: 00000000_H**

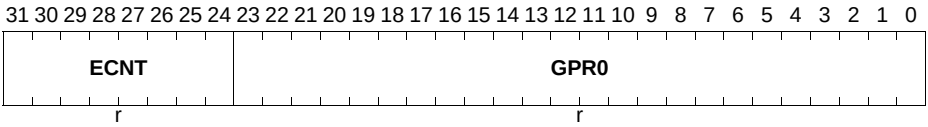


Field	Bits	Type	Description
FLT_FE	[23:0]	rw	**Filter parameter for falling edge** Note: This register has different meanings in the various filter modes. Immediate edge propagation mode = acceptance time for falling edge Individual deglitch time mode = deglitch time for falling edge
Reserved	[31:24]	r	**Reserved** Read as zero, should be written as zero

24.5.7.4 Register TIM0_CHx_GPR0 (x:0…7)

GTM_TIM0_CHx_GPR0 (x=0-7)
TIM0 Channel x General Purpose 0 Register

(01000_H+x*80_H) **Reset Value: $0X000000_H$**

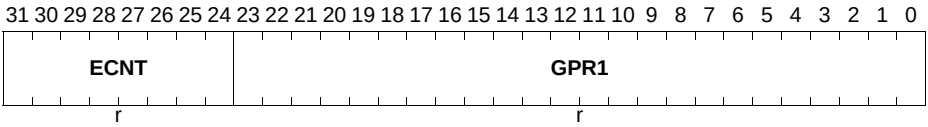


Field	Bits	Type	Description
GPR0	[23:0]	r	**Input signal characteristic parameter 0** Note: The content of this register has different meaning for the TIM channels modes. The content directly depends on the bit fields EGPR0_SEL, GPR0_SEL of register TIM0_CHx_CTRL.
ECNT	[31:24]	r	**Edge counter** Note: The ECNT counts every incoming filtered edge (rising and falling). The counter value is uneven in case of detected rising, and even in case of detected falling edge. Thus, the input signal level is part of the counter and can be obtained by bit 0 of ECNT. Note: The ECNT register is reset to its initial value when the channel is enabled. Please note, that bit 0 depends on the input level coming from the filter unit and defines the reset value immediately.

24.5.7.5 Register TIM0_CHx_GPR1 (x:0…7)

GTM_TIM0_CHx_GPR1 (x=0-7)
TIM0 Channel x General Purpose 1 Register

(01004_H+x*80_H) **Reset Value: $0X000000_H$**

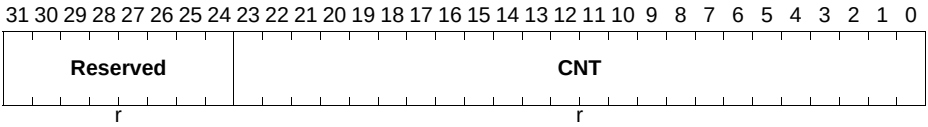


Field	Bits	Type	Description
GPR1	[23:0]	r	**Input signal characteristic parameter 1** Note: The content of this register has different meaning for the TIM channels modes. The content directly depends on the bit fields EGPR1_SEL, GPR1_SEL of register TIM0_CHx_CTRL. Note: In TBCM mode if EGPR1_SEL=1, GPR1_SEL=01 then TIM_INP_VAL is used as input in all other cases TIM filter F_OUT is used as input and bits GPR1(23:8)=0.
ECNT	[31:24]	r	**Edge counter** Note: The ECNT counts every incoming filtered edge (rising and falling). The counter value is uneven in case of detected rising, and even in case of detected falling edge. Thus, the input signal level is part of the counter and can be obtained by bit 0 of ECNT. Note: The ECNT register is reset to its initial value when the channel is enabled. Please note, that bit 0 depends on the input level coming from the filter unit and defines the reset value immediately.

24.5.7.6 Register TIM0_CHx_CNT (x:0…7)

GTM_TIM0_CHx_CNT (x=0-7)
TIM0 Channel x SMU Counter Register

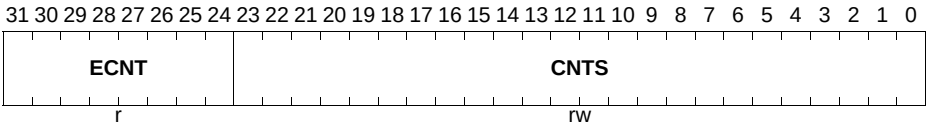
(01008_H+x*80_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
CNT	[23:0]	r	**Actual SMU counter value** Note: The meaning of this value depends on the configured mode: TPWM = actual duration of PWM signal. TPIM = actual duration of all pulses (sum of pulses). TIEM = actual number of received edges. TIPM = actual number of received edges.
Reserved	[31:24]	r	**Reserved** Read as zero, should be written as zero

24.5.7.7 Register TIM0_CHx_CNTS (x:0…7)

GTM_TIM0_CHx_CNTS (x=0-7)
TIM0 Channel x SMU Shadow Counter Register
(01010_H+x*80_H) **Reset Value: $0X000000_H$**

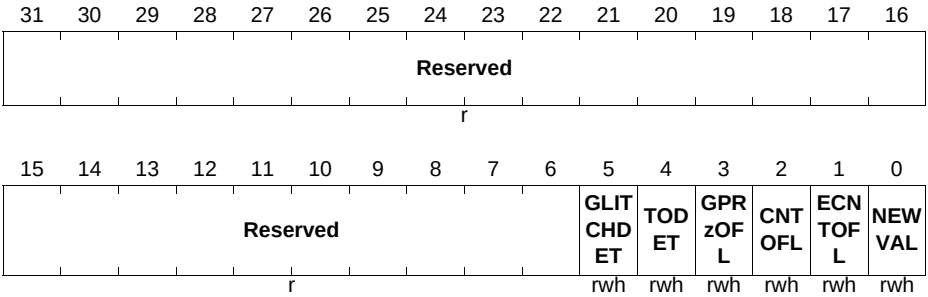


Field	Bits	Type	Description
CNTS	[23:0]	rw	**Counter shadow register** Note: The content of this register has different meaning for the TIM channels modes. The content depends directly on the bit field CNTS_SEL of register TIM0_CHx_CTRL. Note: The register TIM0_CHx_CNTS is only writable in TIPM, TBCM and TGPS mode.
ECNT	[31:24]	r	**Edge counter** Note: The ECNT counts every incoming filtered edge (rising and falling). The counter value is uneven in case of detected rising, and even in case of detected falling edge. Thus, the input signal level is part of the counter and can be obtained by bit 0 of ECNT. Note: The ECNT register is reset to its initial value when the channel is enabled. Please note, that bit 0 depends on the input level coming from the filter unit and defines the reset value immediately.

24.5.7.8 Register TIM0_CHx_IRQ_NOTIFY (x:0…7)

GTM_TIM0_CHx_IRQ_NOTIFY (x=0-7)
TIM0 Channel x Interrupt Notification Register

(0102C$_H$+x*80$_H$) **Reset Value: 00000000$_H$**



Field	Bits	Type	Description
NEWVAL	0	rwh	**New measurement value detected by in channel x (x:0…7)** 0_B No event was occurred 1_B NEWVAL was occurred on the TIM channel Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.
ECNTOFL	1	rwh	**counter overflow of channel x, (x:0…7)** See bit 0
CNTOFL	2	rwh	**SMU CNT counter overflow of channel x, (x:0…7)** See bit 0
GPRzOFL	3	rwh	**data overflow, old data not read out before new data has arrived at input pin, (x:0…7)** See bit 0
TODET	4	rwh	**Timeout reached for input signal of channel x, (x:0…7)** see bit 0
GLITCHDET	5	rwh	**Glitch detected on channel x, (x:0…7)** 0_B no glitch detected for last edge 1_B glitch detected for last edge Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Field	Bits	Type	Description
Reserved	[31:6]	r	**Reserved** Read as zero, should be written as zero

24.5.7.9 Register TIM0_CHx_IRQ_EN (x:0…7)

GTM_TIM0_CHx_IRQ_EN (x=0-7)
TIM0 Channel x Interrupt Enable Register

(01030_H+x*80_H) **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										**GLITCHDET_IRQ_EN**	**TODET_IRQ_EN**	**GPRzOFL_IRQ_EN**	**CNTOFL_IRQ_EN**	**ECNTOFL_IRQ_EN**	**NEWVAL_IRQ_EN**
r										rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
NEWVAL_IRQ_EN	0	rw	**TIM_NEWVALx_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
ECNTOFL_IRQ_EN	1	rw	**TIM_ECNTOFLx_IRQ interrupt enable** see bit 0
CNTOFL_IRQ_EN	2	rw	**TIM_CNTOFLx_IRQ interrupt enable** see bit 0
GPRzOFL_IRQ_EN	3	rw	**TIM_GPRzOFLx_IRQ interrupt enable** see bit 0
TODET_IRQ_EN	4	rw	**TIM_TODETx_IRQ interrupt enable** see bit 0
GLITCHDET_IRQ_EN	5	rw	**TIM_GLITCHDETx_IRQ interrupt enable** see bit 0

Field	Bits	Type	Description
Reserved	[31:6]	r	**Reserved** Read as zero, should be written as zero

24.5.7.10 Register TIM0_CHx_IRQ_FORCINT (x:0…7)

GTM_TIM0_CHx_IRQ_FORCINT (x=0-7)
TIM0 Channel x Software Interrupt Force Register
(01034_H+x*80_H) **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										**TRG_GLITCHDET**	**TRG_TODET**	**TRG_GPRzOFL**	**TRG_CNTOFL**	**TRG_ECNTOFL**	**TRG_NEWVAL**
r										w	w	w	w	w	w

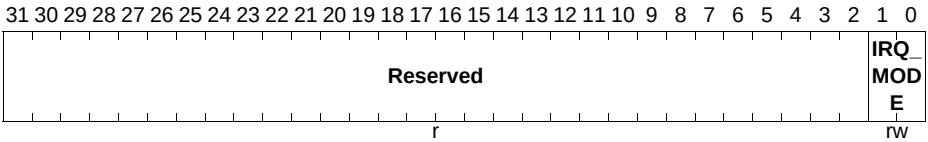
Field	Bits	Type	Description
TRG_NEWVAL	0	w	**Trigger NEWVAL bit in TIM_CHx_IRQ_NOTIFY register by software** 0_B No interrupt triggering 1_B Assert corresponding field in TIM0_CHx_IRQ_NOTIFY register Note: This bit is cleared automatically after write. Note: This bit is write protected by bit GTM_CTRL.RF_PROT.
TRG_ECNTOFL	1	w	**Trigger ECNTOFL bit in TIM_CHx_IRQ_NOTIFY register by software** see bit 0
TRG_CNTOFL	2	w	**Trigger CNTOFL bit in TIM_CHx_IRQ_NOTIFY register by software** see bit 0
TRG_GPRzOFL	3	w	**Trigger GPRzOFL bit in TIM_CHx_IRQ_NOTIFY register by software** see bit 0

Field	Bits	Type	Description
TRG_TODET	4	w	**Trigger TODET bit in TIM_CHx_IRQ_NOTIFY register by software** see bit 0
TRG_GLITCHDET	5	w	**Trigger GLITCHDET bit in TIM_CHx_IRQ_NOTIFY register by software** see bit 0
Reserved	[31:6]	r	**Reserved** Read as zero, should be written as zero

24.5.7.11 Register TIM0_CHx_IRQ_MODE (x:0…7)

GTM_TIM0_CHx_IRQ_MODE (x=0-7)
TIM0 IRQ Mode Configuration Register

(01038_H+x*80_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
IRQ_MODE	[1:0]	rw	**IRQ mode selection** 00_B Level mode 01_B Pulse mode 10_B Pulse-Notify mode 11_B Single-Pulse mode Note: The interrupt modes are described in **Section 24.2.4**.
Reserved	[31:2]	r	**Reserved** Read as zero, should be written as zero

24.5.7.12 Register TIM0_RST

GTM_TIM0_RST

TIM0 Global Software Reset Register(0107C$_H$) **Reset Value: 00000000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RST_CH7	RST_CH6	RST_CH5	RST_CH4	RST_CH3	RST_CH2	RST_CH1	RST_CH0
r								w	w	w	w	w	w	w	w

Field	Bits	Type	Description
RST_CH0	0	w	**Software reset of channel 0** 0_B No action 1_B Reset channel 0 Note: This bit is cleared automatically after write by CPU. The channel registers are set to their reset values and channel operation is stopped immediately.
RST_CH1	1	w	**Software reset of channel 1** see bit 0
RST_CH2	2	w	**Software reset of channel 2** see bit 0
RST_CH3	3	w	**Software reset of channel 3** see bit 0
RST_CH4	4	w	**Software reset of channel 4** see bit 0
RST_CH5	5	w	**Software reset of channel 5** see bit 0
RST_CH6	6	w	**Software reset of channel 6** see bit 0
RST_CH7	7	w	**Software reset of channel 7** see bit 0
Reserved	[31:8]	r	**Reserved** Read as zero, should be written as zero

24.5.7.13 Register TIM0_IN_SRC

GTM_TIM0_IN_SRC
TIM0_IN_SRC Long Name **(01078_H)** **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE_7		VAL_7		MODE_6		VAL_6		MODE_5		VAL_5		MODE_4		VAL_4	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE_3		VAL_3		MODE_2		VAL_2		MODE_1		VAL_1		MODE_0		VAL_0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
VAL_0	[1:0]	rw	**Value to be fed to Channel 0** 00_B Input_signal 0 (ignore write access) 01_B Input_signal is set to 0 01_B Input signal is set to 1 01_B Input signal 1 (ignore write access) *Note: Any read access to a VAL_x, bit field will always result in a value 00 or 11 indicating current state. A modification of the state is only performed with the values 01 and 10. Writing the values 00 and 11 is always ignored.*

Field	Bits	Type	Description
MODE_0	[3:2]	rw	**Input source to Channel 0** 00_B Input Port pin selected by VAL_x (TIM_AUX_IN) or CICTRL is connected to channel (ignore write access) 01_B Input source is set to 0 10_B Input source is set to 1 (VAL_0 is connected to channel) 11_B Input source is defined by VAL_0 (ignore write access) *Note: Note: Any read access to a MODE_x, bit field will always result in a value 00 or 11 indicating current state. A modification of the state is only performed with the values 01 and 10. Writing the values 00 and 11 is always ignored.* *Note: Note: The combination MODE_n=00, VAL_n=11 is allowed. It will be used to route an additional input source TIM_AUX_IN to the desired TIM channel.*
VAL_1	[5:4]	rw	**Value to be fed to Channel 1** see bits 1:0
MODE_1	[7:6]	rw	**Input source to Channel 1** see bits 3:2
VAL_2	[9:8]	rw	**Value to be fed to Channel 2** see bits 1:0
MODE_2	[11:10]	rw	**Input source to Channel 2** see bits 3:2
VAL_3	[13:12]	rw	**Value to be fed to Channel 3** see bits 1:0
MODE_3	[15:14]	rw	**Input source to Channel 3** see bits 3:2
VAL_4	[17:16]	rw	**Value to be fed to Channel 4** see bits 1:0
MODE_4	[19:18]	rw	**Input source to Channel 4** see bits 3:2
VAL_5	[21:20]	rw	**Value to be fed to Channel 5** see bits 1:0

Field	Bits	Type	Description
MODE_5	[23:22]	rw	**Input source to Channel 5** see bits 3:2
VAL_6	[25:24]	rw	**Value to be fed to Channel 6** see bits 1:0
MODE_6	[27:26]	rw	**Input source to Channel 6** see bits 3:2
VAL_7	[29:28]	rw	**Value to be fed to Channel 7** see bits 1:0
MODE_7	[31:30]	rw	**Input source to Channel 7** see bits 3:2

24.5.7.14 Register TIM0_CHx_EIRQ_EN (x:0…7)

GTM_TIM0_CHx_EIRQ_EN (x=0-7)
TIM0 Channel x Error Interrupt Enable Register

($0103C_H$+x*80_H) **Reset Value: 00000000_H**

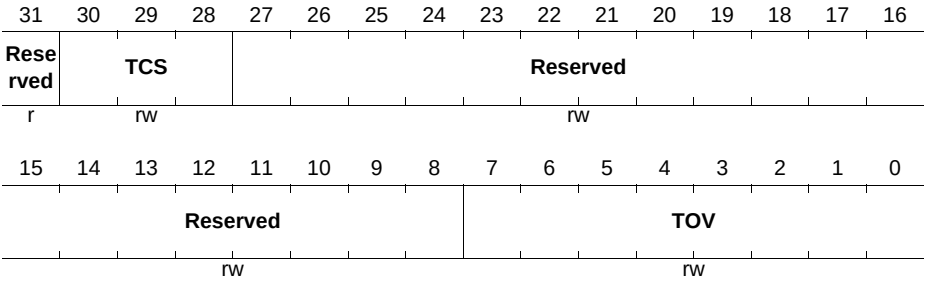
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										GLITCHDET_EIRQ_EN	TODET_EIRQ_EN	GPRzOFL_EIRQ_EN	CNTOFL_EIRQ_EN	ECNTOFL_EIRQ_EN	NEWVAL_EIRQ_EN
r										rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
NEWVAL_EIRQ_EN	0	rw	**TIM_NEWVALx_EIRQ error interrupt enable** 0_B Disable error interrupt, error interrupt is not visible outside GTM 1_B Enable error interrupt, error interrupt is visible outside GTM
ECNTOFL_EIRQ_EN	1	rw	**TIM_ECNTOFLx_IRQ interrupt enable** see bit 0
CNTOFL_EIRQ_EN	2	rw	**TIM_CNTOFLx_IRQ interrupt enable** see bit 0
GPRzOFL_EIRQ_EN	3	rw	**TIM_GPRzOFL_IRQ interrupt enable** see bit 0
TODET_EIRQ_EN	4	rw	**TIM_TODETx_IRQ interrupt enable** see bit 0
GLITCHDET_EIRQ_EN	5	rw	**TIM_GLITCHDETx_IRQ interrupt enable** see bit 0
Reserved	[31:6]	r	**Reserved** Read as zero, should be written as zero

24.5.7.15 Register TIM0_CHx_TDUV (x:0…7)

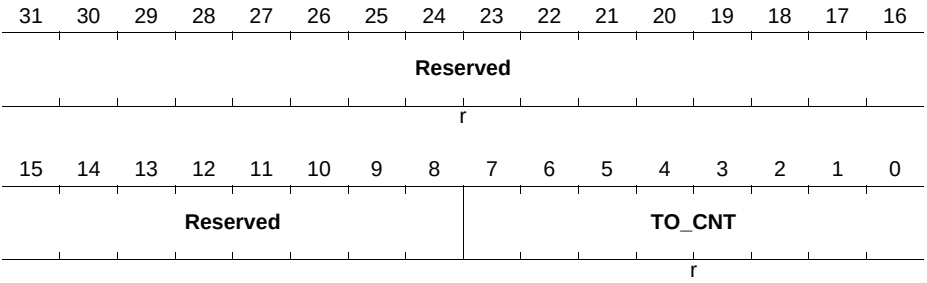
GTM_TIM0_CHx_TDUV (x=0-7)
TIM0 Channel x TDUV Register (01018_H+x*80_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
TOV	[7:0]	rw	**Time out duration for channel x**
Reserved	[27:8]	rw	**Reserved** Read as zero, should be written as zero
TCS	[30:28]	rw	**Timeout Clock selection** 000_B CMU_CLK0 selected 001_B CMU_CLK1 selected 010_B CMU_CLK2 selected 011_B CMU_CLK3 selected 100_B CMU_CLK4 selected 101_B CMU_CLK5 selected 110_B CMU_CLK6 selected 111_B CMU_CLK7 selected
Reserved	31	r	**Reserved** Read as zero, should be written as zero

24.5.7.16 Register TIM0_CHx_TDUC (x:0…7)

GTM_TIM0_CHx_TDUC (x=0-7)
TIM0 Channel x TDUC Register (01014_H+x*80_H) **Reset Value: 00000000_H**

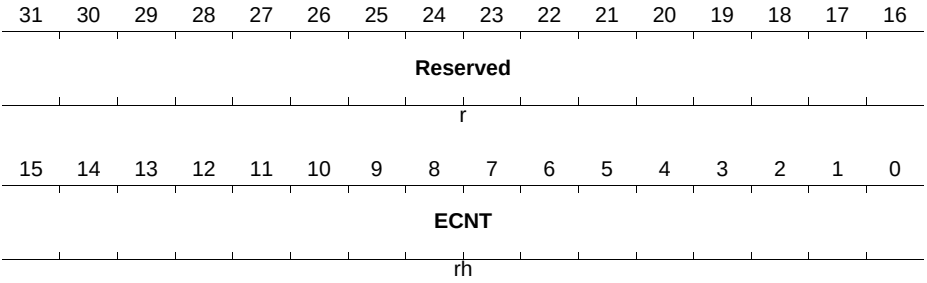


Field	Bits	Type	Description
TO_CNT	[7:0]	rh	**Current Timeout value for channel x**
Reserved	[31:8]	r	**Reserved** Read as zero, should be written as zero

24.5.7.17 Register TIM0_CHx_ECNT (x:0…7)

GTM_TIM0_CHx_ECNT (x=0-7)
TIM0 Channel x Edge Counter Register

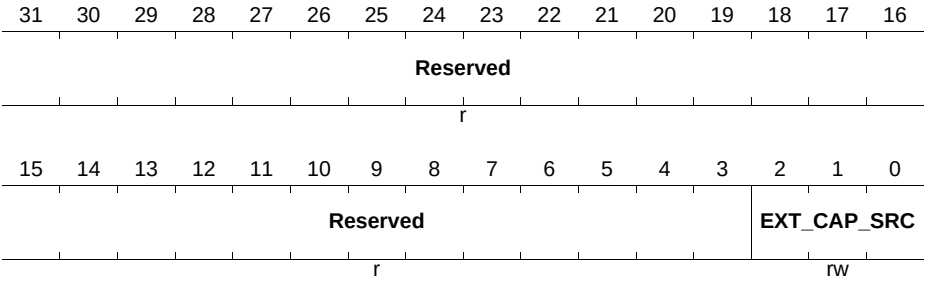
(0100C$_H$+x*80$_H$) **Reset Value: 00000000$_H$**



Field	Bits	Type	Description
ECNT	[15:0]	rh	**Edge counter** Note: If TIM channel is disabled the content of ECNT gets frozen. A read will auto clear the bits [15:1]. Further read access to ECNT will show on Bit 0 the actual input signal value of the channel.
Reserved	[31:16]	r	**Reserved** Read as zero, should be written as zero

24.5.7.18 Register TIM0_CHx_ECTRL (x:0…7)

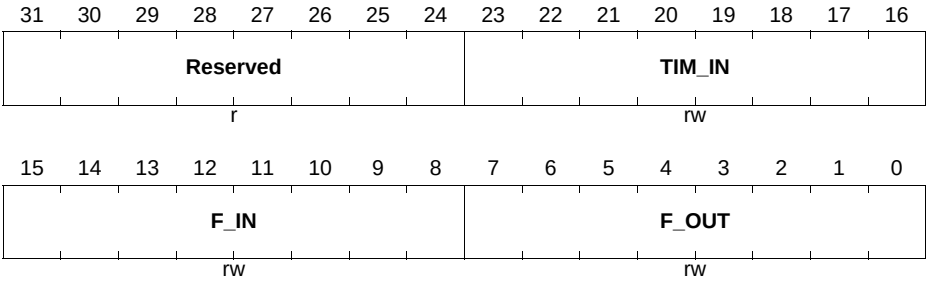
GTM_TIM0_CHx_ECTRL (x=0-7)
TIM0 Channel x External Capture Control Register **(01028_H+x*80_H)** **Reset Value: 00000000_H**



Field	Bits	Type	Description
EXT_CAP_SRC	[2:0]	rw	**Defines selected source for triggering the EXT_CAPTURE functionality** *Note: This register is in use when mode EXT_CAP_EN=1 is configured.* 000_B NEW_VAL_IRQ of previous channel selected 001_B AUX_IN selected 010_B CNTOFL_IRQ of previous channel selected 011_B and CICTRL=1: use signal TIM_IN(x) as input for channel x and CICTRL=0: use signal TIM_IN(x-1) as input for channel x (or TIM_IN(m-1) if x is 0) 100_B ECNTOFL_IRQ of previous channel selected 101_B TODET_IRQ of previous channel selected 110_B GLITCHDET_IRQ of previous channel selected 111_B GPRzOF_IRQof previous channel selected
Reserved	[31:3]	r	**Reserved** Read as zero, should be written as zero

24.5.7.19 Register TIM0_INP_VAL

GTM_TIM0_INP_VAL
TIM0 Input Value Register **(01074_H)** **Reset Value: 00000000_H**



Field	Bits	Type	Description
F_OUT	[7:0]	rw	**Signals after TIM filter unit**
F_IN	[15:8]	rw	**Signals after TINPSRC selection; before TIM filter unit**
TIM_IN	[23:16]	rw	**Signals after TIM input signal synchronisation**
Reserved	[31:24]	r	**Reserved** Read as zero, should be written as zero

24.6 Timer Output Module (TOM)

24.6.1 Overview

The Timer Output Module (TOM) offers 16 independent channels (index x) to generate simple PWM signals at each output pin TOM[i]_CH[x]_OUT.

Additionally, at TOM output TOM[i]_CH15_OUT a pulse count modulated signal can be generated.

The architecture of the TOM submodule is depicted in **Figure 24-28**.

Some useful indices are defined as:

y=0,1

z=0…7

24.6.1.1 TOM Block diagram

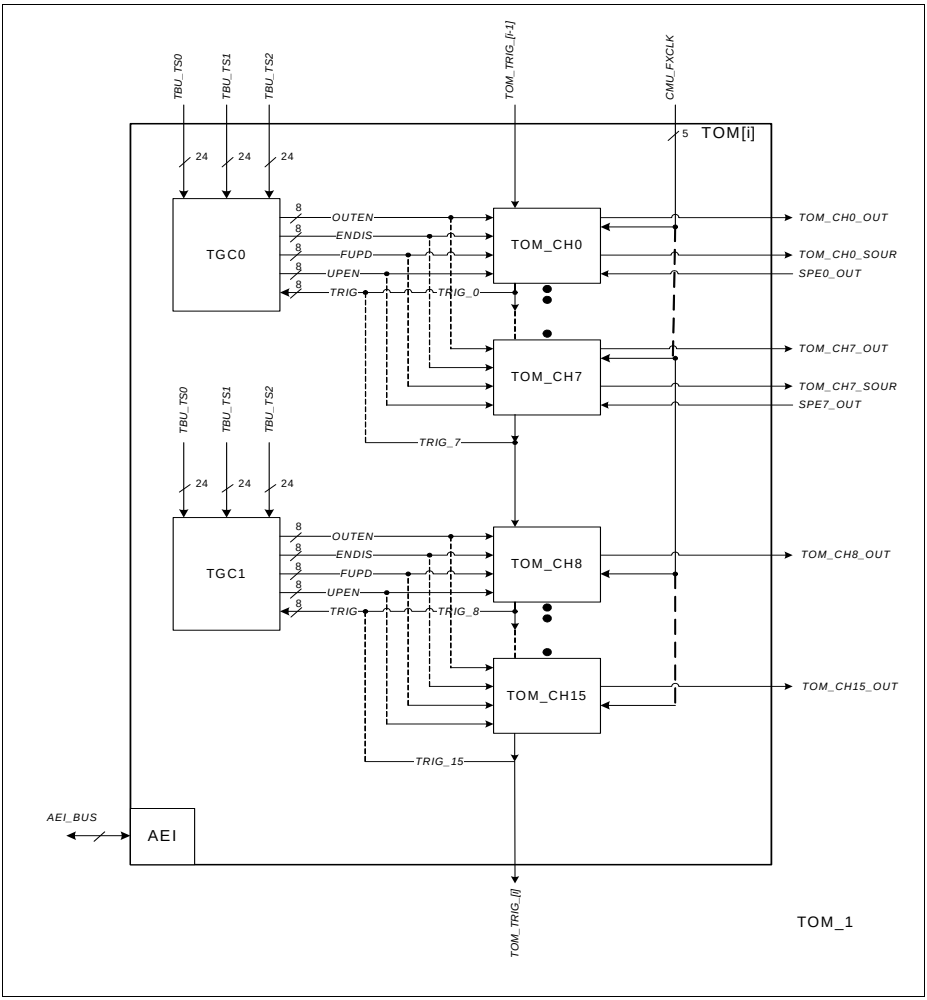


Figure 24-28 TOM Block diagram

The two submodules TGC0 and TGC1 are global channel control units that control the enabling/disabling of the channels and their outputs as well as the update of their period and duty cycle register.

The module TOM receives two (three) timestamp values TBU_TS0, TBU_TS1 (and TBU_TS2) in order to realize synchronized output behaviour on behalf of a common time base.

The 5 dedicated clock line inputs CMU_FXCLK are providing divided clocks that can be selected to clock the output pins.

The trigger signals TOM_TRIG_[i-1] of TOM instance i comes from the preceding instance i-1, the trigger TOM_TRIG_[i] is routed to succeeding instance i+1.

Note: TOM0 is connected to its own output TOMTRIG_0 i.e. the last channel of TOM instance 0 can trigger the first channel of TOM0 instance (this path is registered, which means delayed by one SYS_CLK period).

24.6.2 TOM Global Channel Control (TGC0, TGC1)

24.6.2.1 Overview

There exist two global channel control units (TGC0 and TGC1) to drive a number of individual TOM channels synchronously by external or internal events.

Each TGC[y] can drive up to eight TOM channels where TGC0 controls TOM channels 0 to 7 and TGC1 controls TOM channels 8 to 15.

The TOM submodule supports four different kinds of signalling mechanisms:

- Global enable/disable mechanism for each TOM channel with control register TOMi_TGCy_ENDIS_CTRL and status register TOMi_TGCy_ENDIS_STAT
- Global output enable mechanism for each TOM channel with control register TOMi_TGCy_OUTEN_CTRL and status register TOMi_TGCy_OUTEN_STAT
- Global force update mechanism for each TOM channel with control register TOMi_TGCy_FUPD_CTRL
- Update enable of the register CM0, CM1 and CLK_SRC_STAT for each TOM channel with the control bit field UPEN_CTRL[z] of TOMi_TGCy_GLB_CTRL

24.6.2.2 TGC Subunit

Each of the first three individual mechanisms (enable/disable of the channel, output enable and force update) can be driven by three different trigger sources.

The three trigger sources are:

- the host CPU (bit HOST_TRIG of register TOMi_TGCy_GLB_CTRL)
- the TBU time stamp (signal TBU_TS0, TBU_TS1, TBU_TS2)
- the internal trigger signal TRIG (bunch of trigger signals TRIG_[x]) which can be either the trigger TRIG_CCU0 of channel x, the trigger of preceding channel x-1 (i.e. signal TRIG_[x-1]) or the external trigger TIM_EXT_CAPTURE[t] of assigned TIM channel t.

The first way is to trigger the control mechanism by a direct register write access via host CPU (bit HOST_TRIG of register TOMi_TGCy_GLB_CTRL).

The second way is provided by a compare match trigger on behalf of a specified time base coming from the module TBU (selected by bits TBU_SEL) and the time stamp compare value defined in the bit field ACT_TB of register TOMi_TGCy_ACT_TB.

Note, a signed compare of ACT_TB and selected TBU_TSx with x=0,1,2 is performed.

The third possibility is the input TRIG (bunch of trigger signals TRIG_[x]) coming from the TOM channels 0 to 7 / 8 to 15.

The corresponding trigger signal TRIG_[x] coming from channel [x] can be masked by the register TOMi_TGCy_INT_TRIG.

To enable or disable each individual TOM channel, the registers TOMi_TGCy_ENDIS_CTRL and/or TOMi_TGCy_ENDIS_STAT have to be used.

The register TOMi_TGCy_ENDIS_STAT controls directly the signal ENDIS. A write access to this register is possible.

The register TOMi_TGCy_ENDIS_CTRL is a shadow register that overwrites the value of register TOMi_TGCy_ENDIS_STAT if one of the three trigger conditions matches.

TOM Global channel control mechanism

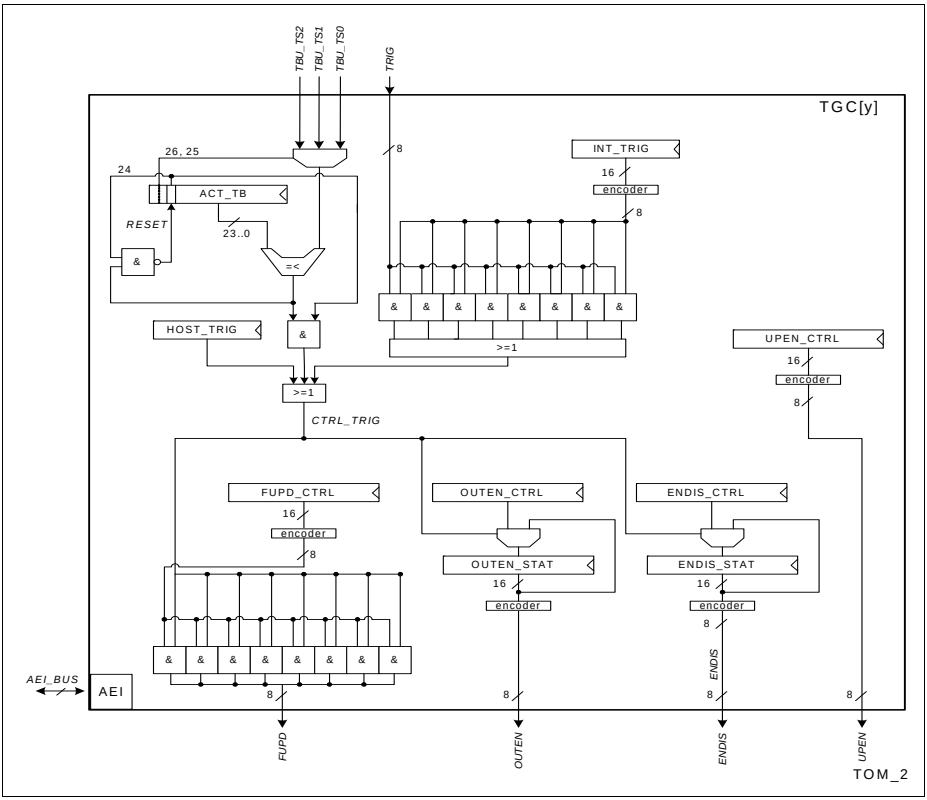


Figure 24-29 TOM Global channel control mechanism

The output of the individual TOM channels can be controlled using the register TOMi_TGCy_OUTEN_CTRL and TOMi_TGCy_OUTEN_STAT.

The register TOMi_TGCy_OUTEN_STAT controls directly the signal OUTEN. A write access to this register is possible.

The register TOMi_TGCy_OUTEN_CTRL is a shadow register that overwrites the value of register TOMi_TGCy_OUTEN_STAT if one of the three trigger conditions matches.

If a TOM channel is disabled by the register TOMi_TGCy_OUTEN_STAT, the actual value of the channel output at TOM_CH[x]_OUT is defined by the signal level bit (SL) defined in the channel control register TOMi_CHx_CTRL.

If the output is enabled, the output at TOM_CH[x]_OUT depends on value of FlipFlop SOUR.

The register TOMi_TGCy_FUPD_CTRL defines which of the TOM channels receive a FORCE_UPDATE event if the trigger signal CTRL_TRIG is raised.

The register bits UPEN_CTRL[z] defines for which TOM channel the update of the working register CM0, CM1 and CLK_SRC by the corresponding shadow register SR0, SR1 and CLK_SRC_SR is enabled. If update is enabled, the register CM0, CM1 and CLK_SRC will be updated on reset of counter register CN0 (see **Figure 24-30** and **Figure 24-31**).

24.6.3 TOM Channel (TOM_CH[x])

Each individual TOM channel comprises a Counter Compare Unit 0 (CCU0), a Counter Compare Unit 1 (CCU1) and the Signal Output Generation Unit (SOU). The architecture is depicted in **Figure 24-30** for channels 0 to 7 and in **Figure 24-31** for channels 8 to 15.

24.6.3.1 TOM Channel 0…7 architecture

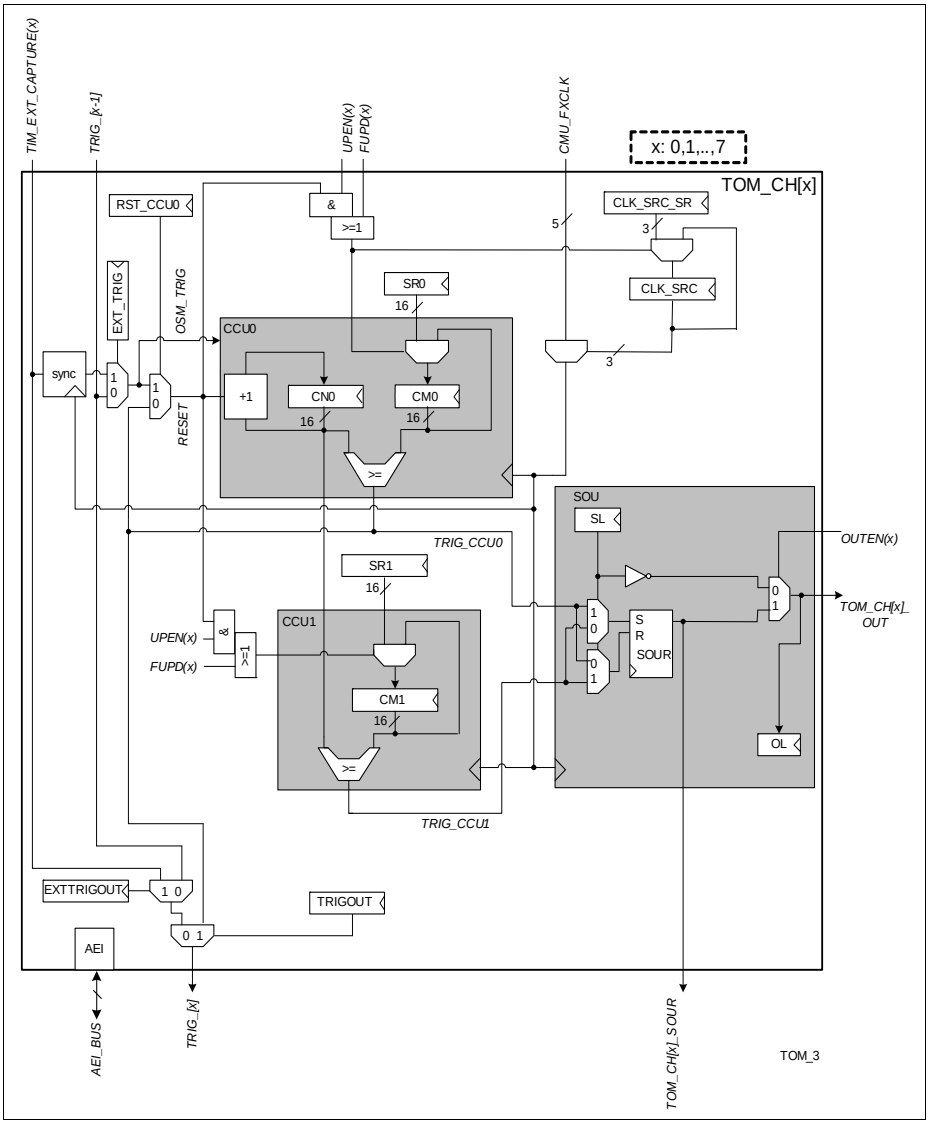


Figure 24-30 TOM Channel 0…7 architecture

24.6.3.2 TOM Channel 8…14 architecture

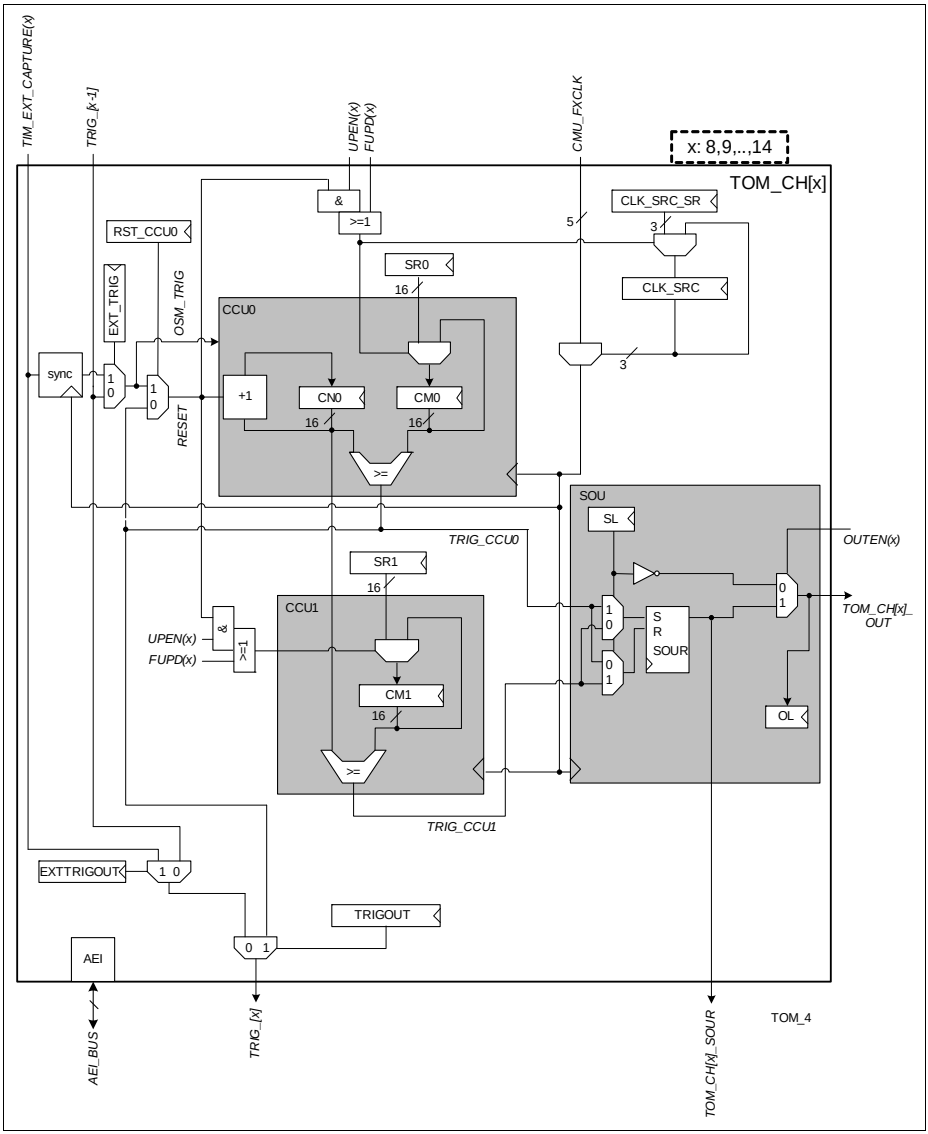


Figure 24-31 TOM Channel 8…14 architecture

24.6.3.3 TOM Channel 15 architecture for PCM generation

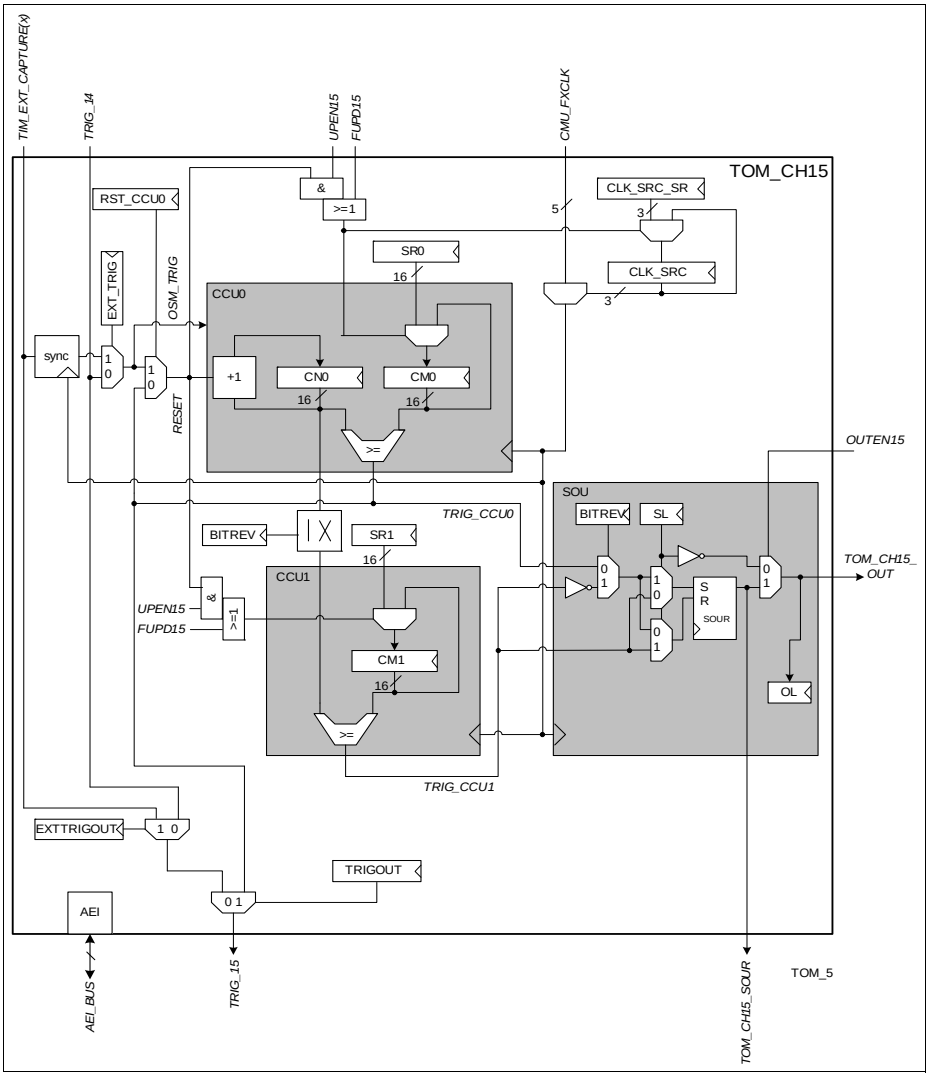


Figure 24-32 TOM Channel 15 architecture for PCM generation

The CCU0 contains a counter CN0 which is clocked with one of the selected input frequencies (CMU_FXCLK) provided from outside of the submodule.

Depending on configuration bits RST_CCU0 of register TOMi_CHx_CTRL the counter register CN0 can be reset either when the counter value is equal to the compare value CM0 or when signalled by the TOM[i] trigger signal TRIG_[x-1] of the preceding channel [x-1] (which can also be the last channel of preceding instance TOM[i-1] or the trigger signal TIM_EXT_CAPTURE(x) of the assigned TIM channel[x].

Note: As an exception, the input TRIG_[0] of instance TOM0 is triggered by its own last channel cCTO via signal TRIG_[15].

When the counter register CN0 is greater or equal than the register CM0, the subunit CCU0 triggers the SOU subunit and the succeeding TOM submodule channel (signal TRIG_CCU0).

In the subunit CCU1 the counter register CN0 is compared with the value of register CM1. If CN0 is greater or equal than CM1 the subunit CCU1 triggers the SOU subunit (signal TRIG_CCU1).

If counter register CN0 of channel x is reset by its own CCU0 unit (i.e. the compare match of CN0>=CM0 configured by RST_CCU0=0), following statements are valid.

- if CM0=0 or CM0=1, 0% duty cycle (=$\overline{SL}$) is generated independent of CM1. The counter CN0 is not counting.
- the configuration of CM1=0 represents 0% duty cycle (=$\overline{SL}$) at the output
- the configuration of CM1>=CM0 represents 100% duty cycle (=SL)

If counter register CN0 of channel x is reset by the trigger signal coming from another channel or the assigned TIM module (configured by RST_CCU0=1), following statements are valid.

- CM0 defines the edge to SL value, CM1 defines the edge to $\overline{SL}$ value
- if CM0=CM1, the output is 100% SL (CM0 has higher priority)
- if CM0=0 and CM1=MAX, the output is 100% SL (while CN0 counts from 0 to MAX-1 and is then reset by trigger)

The hardware ensures that for both 0% and 100% duty cycle no glitch occurs at the output of the TOM channel.

The SOU subunit is responsible for output signal generation. On a trigger TRIG_CCU0 from subunit CCU0 or TRIG_CCU1 from subunit CCU1 a SR-FlipFlop of subunit SOU is either set or reset. If it is set or reset depends on the configuration bit SL of the control register TOMi_CHx_CTRL. The initial signal output level for the channel is the reverse value of the bit SL.

Figure 24-35 clarifies the PWM output behaviour with respect to the SL bit definition.

The output level on the TOM channel output pin TOM[i]_CH[x]_OUT is captured in bit OL of register TOMi_CHx_STAT.

24.6.3.4 Duty cycle, period and selected counter clock frequency update mechanisms

The two action registers CM0 and CM1 can be reloaded with the content of the shadow registers SR0 and SR1. The register CLK_SRC that determines the clock frequency of the counter register CN0 can be reloaded with its shadow register CLK_SRC_SR (bit field in register TOMi_CHx_CTRL)

The update of the register CM0, CM1 and CLK_SRC with the content of its shadow register is done when the reset of the counter register CN0 is requested (via signal RESET). This reset of CN0 is done if the comparison of CN0 greater or equal than CM0 is true or when the reset is triggered by another TOM channel [x-1] via the signal TRIG_[x-1] or when signalled via the signal TIM_EXT_CAPTURE(x) of the assigned TIM channel[x].

For TOM0 channel 0 TRIG_[-1] is '0' hard coded.

With the update of the register CLK_SRC at the end of a period a new counter CN0 clock frequency can easily be adjusted.

An update of duty cycle, period and counter CN0 clock frequency becoming effective synchronously with start of a new period can easily be reached by performing following steps:

1. disable the update of the action register with the content of the corresponding shadow register by setting the channel specific configuration bit UPEN_CTRL[z] of register TOMi_TGCy_GLB_CTRL to '0'.

2. write new desired values to SR0, SR1, CLK_SRC_SR

3. enable update of the action register by setting the channel specific configuration bit UPEN_CTRL[z] of register TOMi_TGCy_GLB_CTRL to '1'.

synchronous update of duty cycle only

A synchronous update of only the duty cycle can be done by simply writing the desired new value to register SR1 without preceding disable of the update mechanism (as described in the section above). The new duty cycle is then applied in the period following the period where the update of register SR1 was done.

synchronous update of duty cycle

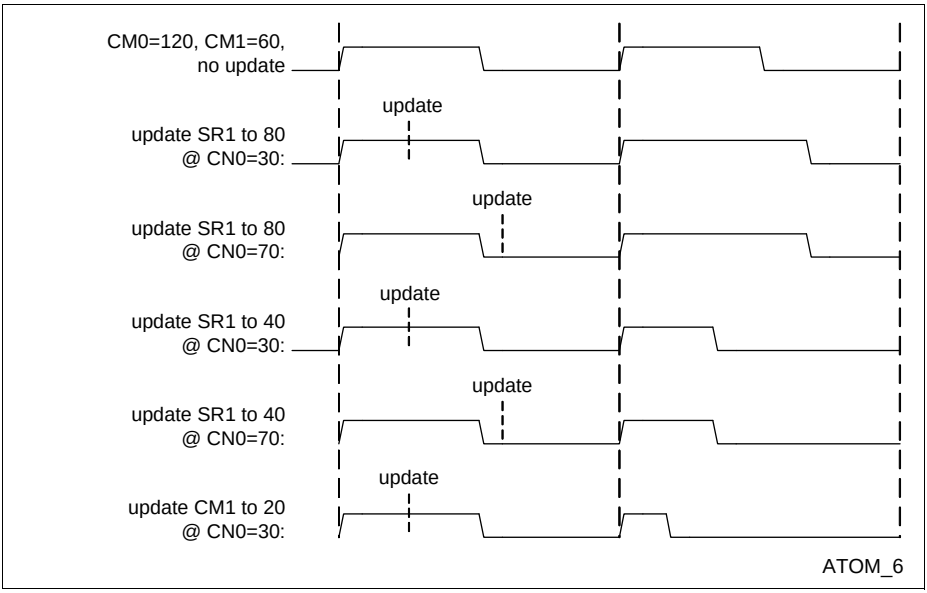


Figure 24-33 synchronous update of duty cycle

asynchronous update of duty cycle only

If the update of the duty cycle should be performed independent of the start of a new period (asynchronous), the desired new value can be written directly to register CM1. In this case it is recommended to additionally either disable the synchronous update mechanism as a whole (i.e. clearing bits UPEN_CTRL[z] of corresponding channel [x] in register TOMi_TGXy_GLB_CTRL) or updating SR1 with the same value as CM1 before writing to CM1.

Depending on the point of time of the update of CM1 in relation to the actual value of CN0 and CM1, the new duty cycle is applied in the current period or the following period (see **Figure 24-34**). In any case the creation of glitches are avoided. The new duty cycle may jitter from update to update by a maximum of one period (given by CM0). However, the period remains unchanged.

asynchronous update of duty cycle

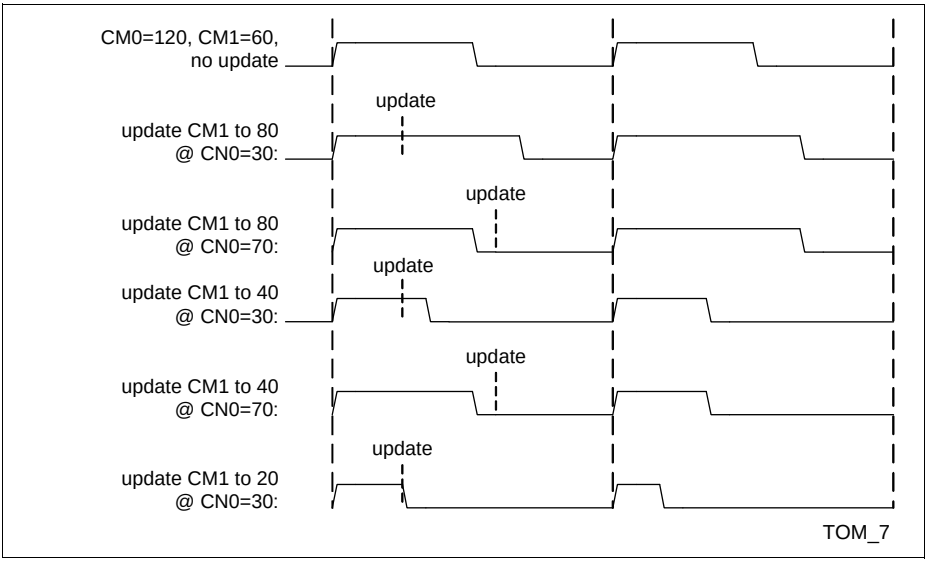


Figure 24-34 asynchronous update of duty cycle

24.6.3.5 TOM continuous mode

In continuous mode the TOM channel starts incrementing the counter register CN0 once it is enabled by setting the corresponding bits in register TOMi_TGCy_ENDIS_STAT (refer to **Section 24.6.2.2** for details of enabling a TOM channel).

The signal level of the generated output signal can be configured with the configuration bit SL of the channel configuration register TOMi_CHx_CTRL.

If the counter CN0 is reset from CM0 back to zero, the first edge of a period is generated at TOM[i]_CH[x]_OUT.

The second edge of the period is generated if CN0 has reached CM1.

Every time the counter CN0 has reached the value of CM0 it is reset back to zero and proceeds with incrementing.

PWM Output with respect to configuration bit

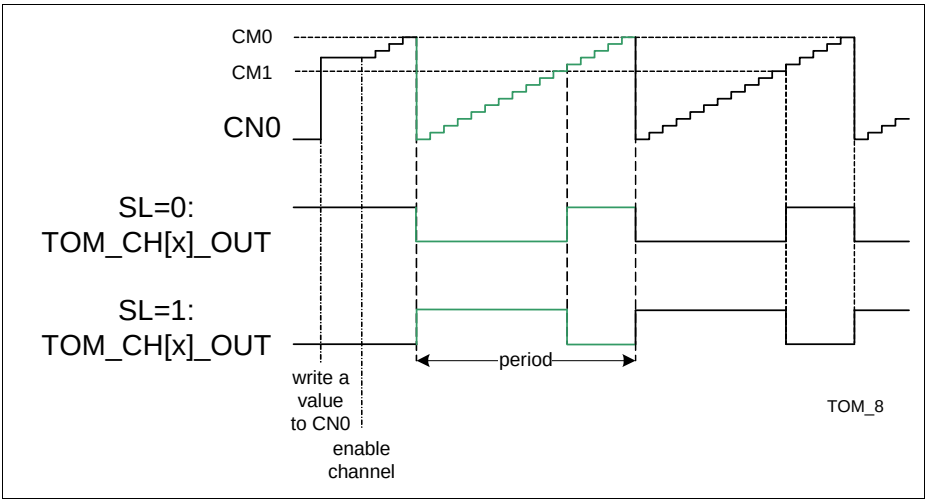


Figure 24-35 PWM Output with respect to configuration bit

24.6.3.6 TOM One shot mode

In One-shot mode, the TOM channel generates one pulse with a signal level specified by the configuration bit SL in the channel [x] configuration register TOMi_CHx_CTRL.

First the channel has to be enabled by setting the corresponding TOMi_TGCy_ENDIS_STAT value and the one-shot mode has to be enabled by setting bit OSM in register TOMi_CHx_CTRL.

In one-shot mode the counter CN0 will not be increment once the channel is enabled.

A write access to the register CN0 triggers the start of pulse generation (i.e. the increment of the counter register CN0).

The new value of CN0 determines the start delay of the first edge. The delay time of the first edge is given by (CM0-CN0) multiplied with period defined by current value of CLK_SRC.

If the counter CN0 is reset from CM0 back to zero, the first edge at TOM[i]_CH[x]_OUT is generated.

To avoid an update of CMx register with content of SRx register at this point in time, the automatic update should be disabled by setting UPEN_CTRLx = 00 (in register TOMi_CHx_CTRL).

The second edge is generated if CN0 is greater or equal than CM1 (i.e. CN0 was increment until it has reached CM1 or CN0 is greater than CM1 after an update of CM1).

If the counter CN0 has reached the value of CM0 a second time, the counter stops.

PWM Output with respect to configuration bit SL in one-shot mode: trigger by writing to CN0

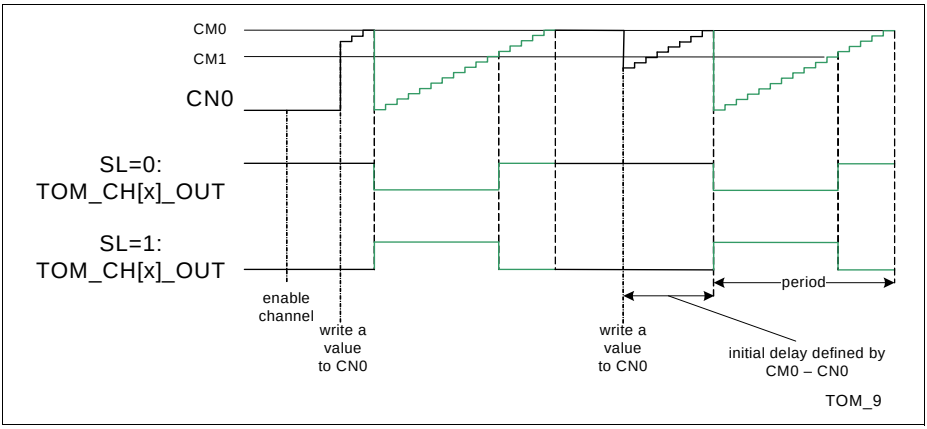


Figure 24-36 PWM Output with respect to configuration bit SL in one-shot mode: trigger by writing to CN0: trigger by writing to CN0

Further output of single periods can be started by a write access to register CN0.

Writing to incrementing counter CN0 a value CN0new < CM1 while CN0old is below CM1 leads to a lengthening of the pulse. The counter CN0 stops if is reaches CM0.

Writing to incrementing counter CN0 a value CN0new > CM1 while CN0old is already greater than CM1 leads to an immediate restart of a single pulse generation inclusive the initial delay defined by CM0 - CN0new.

If a channel is configured to one-shot mode and configuration bit OSM_TRIG is set to 1, the trigger signal OSM_TRIG (i.s. TRIG_[x-1] or TIM_EXT_CAPTURE[x]) triggers start of one pulse generation.

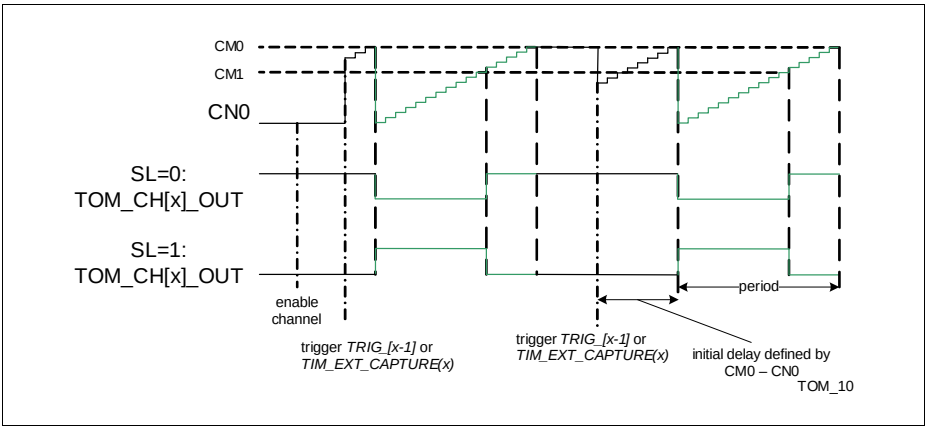


Figure 24-37 PWM Output with respect to configuration bit SL in one-shot mode: trigger by TRIG_[x-1] or TIM_EXT_CAPTURE(x)

24.6.3.7 Pulse count modulation

At the output TOM[i]_CH15_OUT a pulse count modulated signal can be generated instead of the simple PWM output signal.

Figure 24-32 outlines the circuit for Pulse Count Modulation.

The PCM mode is enabled by setting bit BITREV to 1.

With the configuration bit BITREV=1 a bit-reversing of the counter output CN0 is configured. In this case the bits LSB and MSB are swapped, the bits LSB+1 and MSB-1 are swapped, the bits LSB+2 and MSB-2 are swapped and so on.

The effect of bit-reversing of the CN0 register value is shown in the following **Figure 24-38**.

Bit reversing of counter

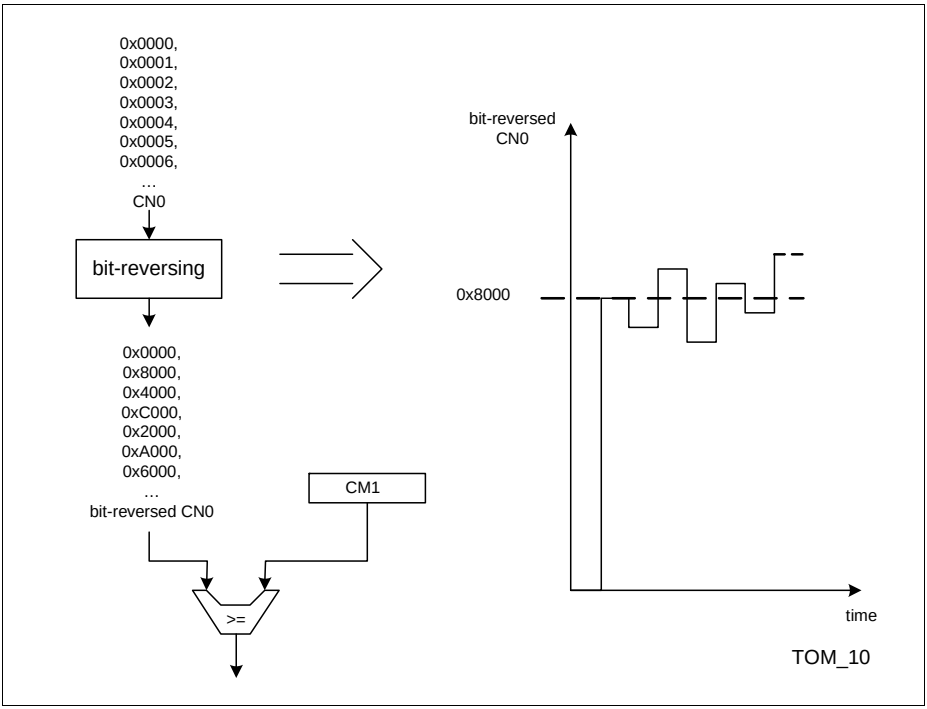


Figure 24-38 Bit reversing of counter

In the PCM mode the counter register CN0 is increment by every clock tick depending on configured CMU clock (CMU_FXCLK).

The output of counter register CN0 is first bit-reversed and than compared with the configured register value CM1.

If the bit-reversed value of register CN0 is greater than CM1, the SR-FlipFlop of submodule SOU is set (depending on configuration register SL) otherwise the SR-FlipFlop is reset. This generates at the output TOM[i]_CH15_OUT a pulse count modulated signal.

In PCM mode the CM0 register always has to be set to its maximum value 0xFFFF.

24.6.4 TOM Interrupt signals

The following table describes TOM interrupt signals:

Table 24-18 TOM Interrupt signals

Signal	Description
TOM_CCU0TCx_IRQ	CCU0 Trigger condition interrupt for channel x
TOM_CCU1TCx_IRQ	CCU1 Trigger condition interrupt for channel x

24.6.5 TOM Configuration Register overview

The following table gives an overview of the TOM configuration registers.

Table 24-19 TOM Configuration Register overview

Register name	Description	Details in Section
TOMi_TGC0_GLB_CTRL	TGC0 global control register	Section 24.6.6.1
TOMi_TGC1_GLB_CTRL	TGC1 global control register	
TOMi_TGC0_ENDIS_CTRL	TGC0 enable/disable control register	Section 24.6.6.2
TOMi_TGC1_ENDIS_CTRL	TGC1 enable/disable control register	
TOMi_TGC0_ENDIS_STAT	TGC0 enable/disable status register	Section 24.6.6.3
TOMi_TGC1_ENDIS_STAT	TGC1 enable/disable status register	
TOMi_TGC0_ACT_TB	TGC0 action time base register	Section 24.6.6.4
TOMi_TGC1_ACT_TB	TGC1 action time base register	
TOMi_TGC0_OUTEN_CTRL	TGC0 output enable control register	Section 24.6.6.5
TOMi_TGC1_OUTEN_CTRL	TGC1 output enable control register	
TOMi_TGC0_OUTEN_STAT	TGC0 output enable status register	Section 24.6.6.6
TOMi_TGC1_OUTEN_STAT	TGC1 output enable status register	
TOMi_TGC0_FUPD_CTRL	TGC0 force update control register	Section 24.6.6.7
TOMi_TGC1_FUPD_CTRL	TGC1 force update control register	

Table 24-19 TOM Configuration Register overview (cont'd)

Register name	Description	Details in Section
TOMi_TGC0_INT_TRIG	TGC0 internal trigger control register	Section 24.6.6.8
TOMi_TGC1_INT_TRIG	TGC1 internal trigger control register	
TOMi_CHx_CTRL	TOM Channel x control register (x=0…14)	Section 24.6.6.9
TOMi_CH15_CTRL	TOM Channel 15 control register	Section 24.6.6.10
TOMi_CHx_CN0	TOM Channel x CCU0 counter register (x=0…15)	Section 24.6.6.11
TOMi_CHx_CM0	TOM Channel x CCU0 compare register (x=0…15)	Section 24.6.6.12
TOMi_CHx_SR0	TOM Channel x CCU0 compare shadow register (x=0…15)	Section 24.6.6.13
TOMi_CHx_CM1	TOM Channel x CCU1 compare register (x=0…15)	Section 24.6.6.14
TOMi_CHx_SR1	TOM Channel x CCU1 compare shadow register (x=0…15)	Section 24.6.6.15
TOMi_CHx_STAT	TOM channel status register (x=0…15)	Section 24.6.6.16
TOMi_CHx_IRQ_NOTIFY	TOM channel x interrupt notification register (x=0…15)	Section 24.6.6.17
TOMi_CHx_IRQ_EN	TOM channel x interrupt enable register (x=0…15)	Section 24.6.6.18
TOMi_CHx_IRQ_FORCINT	TOM channel x software interrupt generation register (x=0…15)	Section 24.6.6.19
TOMi_CHx_IRQ_MODE	IRQ mode configuration register (x=0…15)	Section 24.6.6.20

24.6.6 TOM Configuration Registers Description

All of the following registers are 32-bit only accessible.

24.6.6.1 Register TOMi_TGC0_GLB_CTRL

GTM_TOMi_TGC0_GLB_CTRL (i=0-1)
TOMi TGC0 Global Control Register(08030_H+i*800_H) **Reset Value: 00000000_H**
GTM_TOMi_TGC1_GLB_CTRL (i=0-1)
TOMi TGC1 Global Control Register(08230_H+i*800_H) **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPEN_CTRL7		UPEN_CTRL6		UPEN_CTRL5		UPEN_CTRL4		UPEN_CTRL3		UPEN_CTRL2		UPEN_CTRL1		UPEN_CTRL0	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RST_CH7	RST_CH6	RST_CH5	RST_CH4	RST_CH3	RST_CH2	RST_CH1	RST_CH0	Reserved							HOST_TRIG
w	w	w	w	w	w	w	w	r							w

Field	Bits	Type	Description
HOST_TRIG	0	w	**Trigger request signal (see TGC0, TGC1) to update the register ENDIS_STAT and OUTEN_STAT** 0_B no trigger request 1_B set trigger request Read as 0. *Note:* This flag is cleared automatically after triggering the update
Reserved	[7:1]	r	**Reserved** Read as zero, should be written as zero
RST_CH0	8	w	**Software reset of channel 0** 0_B No action 1_B Reset channel Read as 0. *Note: This bit is cleared automatically after write by CPU. The channel registers are set to their reset values and channel operation is stopped immediately. The S-r FlipFlop SOUR is set to '1'.*

Field	Bits	Type	Description
RST_CH1	9	w	**Software reset of channel 1** See bit 8
RST_CH2	10	w	**Software reset of channel 2** See bit 8
RST_CH3	11	w	**Software reset of channel 3** See bit 8
RST_CH4	12	w	**Software reset of channel 4** See bit 8
RST_CH5	13	w	**Software reset of channel 5** See bit 8
RST_CH6	14	w	**Software reset of channel 6** See bit 8
RST_CH7	15	w	**Software reset of channel 7** See bit 8
UPEN_CTRL0	[17:16]	rw	**TOM channel 0 enable update of register CM0, CM1 and CLK_SRC from SR0, SR1 and CLK_SRC_SR** Write of following double bit values is possible: 00_B don't care, bits 1:0 will not be changed 01_B update disabled: is read as 00 (see below) 10_B update enabled: is read as 11 (see below) 11_B don't care, bits 1:0 will not be changed Read of following double values means: 00_B channel disabled 11_B channel enabled
UPEN_CTRL1	[19:18]	rw	**TOM channel 1 enable update of register CM0, CM1 and CLK_SRC** See bits 17:16
UPEN_CTRL2	[21:20]	rw	**TOM channel 2 enable update of register CM0, CM1 and CLK_SRC** See bits 17:16
UPEN_CTRL3	[23:22]	rw	**TOM channel 3 enable update of register CM0, CM1 and CLK_SRC** See bits 17:16
UPEN_CTRL4	[25:24]	rw	**TOM channel 4 enable update of register CM0, CM1 and CLK_SRC** See bits 17:16

Field	Bits	Type	Description
UPEN_CTRL5	[27:26]	rw	**TOM channel 5 enable update of register CM0, CM1 and CLK_SRC** See bits 17:16
UPEN_CTRL6	[29:28]	rw	**TOM channel 6 enable update of register CM0, CM1 and CLK_SRC** See bits 17:16
UPEN_CTRL7	[31:30]	rw	**TOM channel 7 enable update of register CM0, CM1 and CLK_SRC** See bits 17:16

24.6.6.2 Register TOMi_TGC0_ENDIS_CTRL

GTM_TOMi_TGC0_ENDIS_CTRL (i=0-1)
TOMi TGC0 Enable/Disable Control Register
(08070_H+i*800_H) **Reset Value: 00000000_H**

GTM_TOMi_TGC1_ENDIS_CTRL (i=0-1)
TOMi TGC1 Enable/Disable Control Register
(08270_H+i*800_H) **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENDIS_CTRL7		ENDIS_CTRL6		ENDIS_CTRL5		ENDIS_CTRL4		ENDIS_CTRL3		ENDIS_CTRL2		ENDIS_CTRL1		ENDIS_CTRL0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
ENDIS_CTRL0	[1:0]	rw	**TOM channel 0 enable/disable update value** If a TOM channel is disabled, the counter CN0 is stopped and the FlipFlop SOUR is set to the inverse value of control bit SL. On an enable event, the counter CN0 starts counting from its current value. Write of following double bit values is possible: 00_B don't care, bits 1:0 of register ENDIS_STAT will not be changed on an update trigger 01_B disable channel on an update trigger 10_B enable channel on an update trigger 11_B don't change bits 1:0 of this register Note: if the channel is disabled (ENDIS[0]=0) or the output is disabled (OUTEN[0]=0), the TOM channel 0 output TOM_OUT[0] is the inverted value of bit SL.
ENDIS_CTRL1	[3:2]	rw	**TOM channel 1 enable/disable update value** See bits 1:0
ENDIS_CTRL2	[5:4]	rw	**TOM channel 2 enable/disable update value** See bits 1:0
ENDIS_CTRL3	[7:6]	rw	**TOM channel 3 enable/disable update value** See bits 1:0

Field	Bits	Type	Description
ENDIS_CTRL4	[9:8]	rw	**TOM channel 4 enable/disable update value** See bits 1:0
ENDIS_CTRL5	[11:10]	rw	**TOM channel 5 enable/disable update value** See bits 1:0
ENDIS_CTRL6	[13:12]	rw	**TOM channel 6 enable/disable update value** See bits 1:0
ENDIS_CTRL7	[15:14]	rw	**TOM channel 7 enable/disable update value** See bits 1:0
Reserved	[31:16]	r	**Reserved** Read as zero, should be written as zero

24.6.6.3 Register TOMi_TGC0_ENDIS_STAT

GTM_TOMi_TGC0_ENDIS_STAT (i=0-1)
TOMi TGC0 Enable/Disable Status Register
(08074_H+i*800_H) **Reset Value: 00000000_H**

GTM_TOMi_TGC1_ENDIS_STAT (i=0-1)
TOMi TGC1 Enable/Disable Status Register
(08274_H+i*800_H) **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENDIS_STAT7		ENDIS_STAT6		ENDIS_STAT5		ENDIS_STAT4		ENDIS_STAT3		ENDIS_STAT2		ENDIS_STAT1		ENDIS_STAT0	
rw		rw		rw		rw		rw		rw		rw		rw	

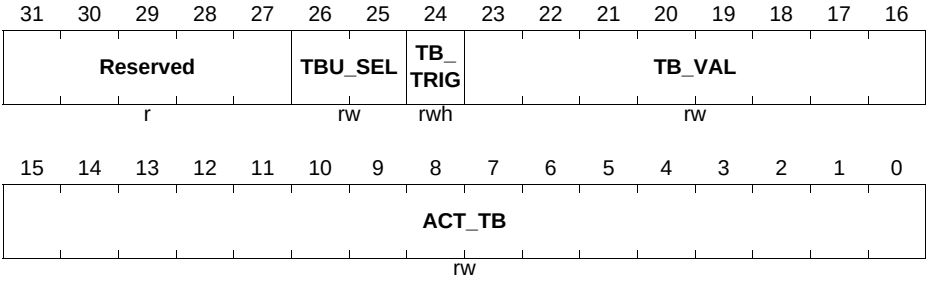
Field	Bits	Type	Description
ENDIS_STAT0	[1:0]	rw	**TOM channel 0 enable/disable** If a TOM channel is disabled, the counter CN0 is stopped and the FlipFlop SOUR is set to the inverse value of control bit SL. On an enable event, the counter CN0 starts counting from its current value. Write of following double bit values is possible: 00_B don't care, bits 1:0 will not be changed 01_B channel disabled: is read as 00 (see below) 10_B channel enabled: is read as 11 (see below) 11_B don't care, bits 1:0 will not be changed Read of following double values means: 00_B channel disable 11_B channel enable
ENDIS_STAT1	[3:2]	rw	**TOM channel 1 enable/disable** See bits 1:0
ENDIS_STAT2	[5:4]	rw	**TOM channel 2 enable/disable** See bits 1:0
ENDIS_STAT3	[7:6]	rw	**TOM channel 3 enable/disable** See bits 1:0

Field	Bits	Type	Description
ENDIS_STAT4	[9:8]	rw	**TOM channel 4 enable/disable** See bits 1:0
ENDIS_STAT5	[11:10]	rw	**TOM channel 5 enable/disable** See bits 1:0
ENDIS_STAT6	[13:12]	rw	**TOM channel 6 enable/disable** See bits 1:0
ENDIS_STAT7	[15:14]	rw	**TOM channel 7 enable/disable** See bits 1:0
Reserved	[31:16]	r	**Reserved** Read as zero, should be written as zero

24.6.6.4 Register TOMi_TGC0_ACT_TB

GTM_TOMi_TGC0_ACT_TB (i=0-1)
TOMi TGC0 Action Time Base Register
(08034_H+i*800_H) **Reset Value: 00000000_H**

GTM_TOMi_TGC1_ACT_TB (i=0-1)
TOMi TGC1 Action Time Base Register
(08234_H+i*800_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
ACT_TB	[23:0]	rw	**Time base value** specifies the signed compare value with selected signal TBU_TCx (x=0…2). If selected TBU_TSx value is in the interval [ACT_TB-$007FFFFF_H$,ACT_TB] the event is in the past and the trigger is generated immediately. Otherwise the event is in the future and the trigger is generated if selected TBU_TSx is equal to ACT_TB.
TB_TRIG	24	rwh	**Set trigger request** 0_B no trigger request 1_B set trigger request Note: This flag is reset automatically if the selected time base unit (TBU_TS0 or TBU_TS1 or TBU_TS2 if present) has reached the value ACT_TB and the update of the register were triggered.

Field	Bits	Type	Description
TBU_SEL	[26:25]	rw	**Selection of time base used for comparison** 00_B TBU_TS0 selected 01_B TBU_TS1 selected 10_B TBU_TS2 selected 11_B Reserved; same as 00_B Note: The bit combination “10” is only applicable if the TBU of the device contains three time base channels. Otherwise, this bit combination is also reserved. Please refer to GTM Architecture block diagram on page 3 to determine the number of channels for TBU of this device.
Reserved	[31:27]	r	**Reserved** Read as zero, should be written as zero

24.6.6.5 Register TOMi_TGC0_OUTEN_CTRL

GTM_TOMi_TGC0_OUTEN_CTRL (i=0-1)
TOMi TGC0 Output Enable Control Register
(08078_H+i*800_H) **Reset Value: 00000000_H**
GTM_TOMi_TGC1_OUTEN_CTRL (i=0-1)
TOMi TGC1 Output Enable Control Register
(08278_H+i*800_H) **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUTEN_CTRL7		OUTEN_CTRL6		OUTEN_CTRL5		OUTEN_CTRL4		OUTEN_CTRL3		OUTEN_CTRL2		OUTEN_CTRL1		OUTEN_CTRL0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
OUTEN_CTRL0	[1:0]	rw	**Output TOM_OUT(0) enable/disable update value** Write of following double bit values is possible: 00_B don't care, bits 1:0 of register OUTEN_STAT will not be changed on an update trigger 01_B disable channel output on an update trigger 10_B enable channel output on an update trigger 11_B don't change bits 1:0 of this register Note: if the channel is disabled (ENDIS[0]=0) or the output is disabled (OUTEN[0]=0), the TOM channel 0 output TOM_OUT[0] is the inverted value of bit SL.
OUTEN_CTRL1	[3:2]	rw	**Output TOM_OUT(1)enable/disable update value** See bits 1:0
OUTEN_CTRL2	[5:4]	rw	**Output TOM_OUT(2) enable/disable update value** See bits 1:0
OUTEN_CTRL3	[7:6]	rw	**Output TOM_OUT(3) enable/disable update value** See bits 1:0
OUTEN_CTRL4	[9:8]	rw	**Output TOM_OUT(4) enable/disable update value** See bits 1:0
OUTEN_CTRL5	[11:10]	rw	**Output TOM_OUT(5) enable/disable update value** See bits 1:0
OUTEN_CTRL6	[13:12]	rw	**Output TOM_OUT(6) enable/disable update value** See bits 1:0
OUTEN_CTRL7	[15:14]	rw	**Output TOM_OUT(7) enable/disable update value** See bits 1:0
Reserved	[31:16]	r	**Reserved** Read as zero, should be written as zero

24.6.6.6 Register TOMi_TGC0_OUTEN_STAT

GTM_TOMi_TGC0_OUTEN_STAT (i=0-1)
TOMi TGC0 Output Enable Status Register
($0807C_H$+i*800_H) **Reset Value: 00000000_H**

GTM_TOMi_TGC1_OUTEN_STAT (i=0-1)
TOMi TGC1 Output Enable Status Register
($0827C_H$+i*800_H) **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUTEN_STAT7		OUTEN_STAT6		OUTEN_STAT5		OUTEN_STAT4		OUTEN_STAT3		OUTEN_STAT2		OUTEN_STAT1		OUTEN_STAT0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
OUTEN_STAT0	[1:0]	rw	**Control/status of output TOM_OUT(0)** Write of following double bit values is possible: 00_B don't care, bits 1:0 will not be changed 01_B channel disabled: is read as 00 (see below) 10_B channel enabled: is read as 11 (see below) 11_B don't care, bits 1:0 will not be changed Read of following double values means: 00_B channel disable 11_B channel enable
OUTEN_STAT1	[3:2]	rw	**Control/status of output TOM_OUT(1)** See bits 1:0
OUTEN_STAT2	[5:4]	rw	**Control/status of output TOM_OUT(2)** See bits 1:0
OUTEN_STAT3	[7:6]	rw	**Control/status of output TOM_OUT(3)** See bits 1:0
OUTEN_STAT4	[9:8]	rw	**Control/status of output TOM_OUT(4)** See bits 1:0
OUTEN_STAT5	[11:10]	rw	**Control/status of output TOM_OUT(5)** See bits 1:0

Field	Bits	Type	Description
OUTEN_STAT6	[13:12]	rw	**Control/status of output TOM_OUT(6)** See bits 1:0
OUTEN_STAT7	[15:14]	rw	**Control/status of output TOM_OUT(7)** See bits 1:0
Reserved	[31:16]	r	**Reserved** Read as zero, should be written as zero

24.6.6.7 Register TOMi_TGC0_FUPD_CTRL

GTM_TOMi_TGC0_FUPD_CTRL (i=0-1)
TOMi TGC0 Force Update Control Register
(08038_H+i*800_H) **Reset Value: 00000000_H**

GTM_TOMi_TGC1_FUPD_CTRL (i=0-1)
TOMi TGC1 Force Update Control Register
(08238_H+i*800_H) **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSTCN0_CH7		RSTCN0_CH6		RSTCN0_CH5		RSTCN0_CH4		RSTCN0_CH3		RSTCN0_CH2		RSTCN0_CH1		RSTCN0_CH0	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FUPD_CTRL7		FUPD_CTRL6		FUPD_CTRL5		FUPD_CTRL4		FUPD_CTRL3		FUPD_CTRL2		FUPD_CTRL1		FUPD_CTRL0	
rw		rw		rw		rw		rw		rw		rw		rw	

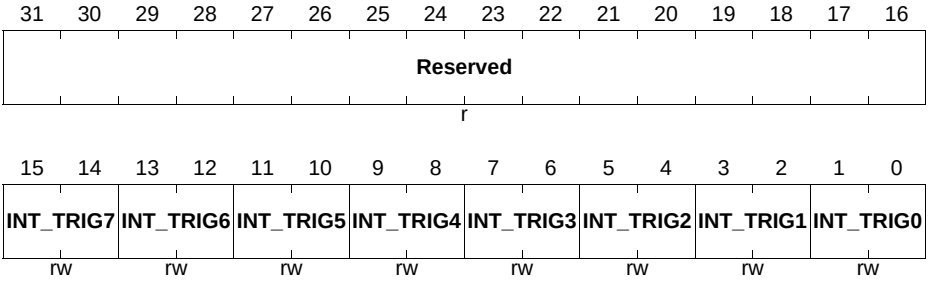
Field	Bits	Type	Description
FUPD_CTRL0	[1:0]	rw	**Force update of TOM channel 0 operation registers** Write of following double bit values is possible: 00_B don't care, bits 1:0 will not be changed 01_B force update disabled: is read as 00 (see below) 10_B force update enabled: is read as 11 (see below) 11_B don't care, bits 1:0 will not be changed Read of following double values means: 00_B force update disabled 11_B force channel enabled
FUPD_CTRL1	[3:2]	rw	**Force update of TOM channel 1 operation registers** See bits 1:0
FUPD_CTRL2	[5:4]	rw	**Force update of TOM channel 2 operation registers** See bits 1:0
FUPD_CTRL3	[7:6]	rw	**Force update of TOM channel 3 operation registers** See bits 1:0
FUPD_CTRL4	[9:8]	rw	**Force update of TOM channel 4 operation registers** See bits 1:0
FUPD_CTRL5	[11:10]	rw	**Force update of TOM channel 5 operation registers** See bits 1:0

Field	Bits	Type	Description
FUPD_CTRL6	[13:12]	rw	**Force update of TOM channel 6 operation registers** See bits 1:0
FUPD_CTRL7	[15:14]	rw	**Force update of TOM channel 7 operation registers** See bits 1:0
RSTCN0_CH0	[17:16]	rw	**Reset CN0 of channel 0 on force update event** Write of following double bit values is possible: 00_B don't care, bits 1:0 will not be changed 01_B CN0 is not reset on forced update: is read as 00 (see below) 10_B CN0 is reset on forced update: is read as 11 (see below) 11_B don't care, bits 1:0 will not be changed Read of following double values means: 00_B CN0 is not reset on forced update 11_B CN0 is reset on forced update
RSTCN0_CH1	[19:18]	rw	**Reset CN0 of channel 1 on force update event** See bits 17:16
RSTCN0_CH2	[21:20]	rw	**Reset CN0 of channel 2 on force update event** See bits 17:16
RSTCN0_CH3	[23:22]	rw	**Reset CN0 of channel 3 on force update event** See bits 17:16
RSTCN0_CH4	[25:24]	rw	**Reset CN0 of channel 4 on force update event** See bits 17:16
RSTCN0_CH5	[27:26]	rw	**Reset CN0 of channel 5 on force update event** See bits 17:16
RSTCN0_CH6	[29:28]	rw	**Reset CN0 of channel 6 on force update event** See bits 17:16
RSTCN0_CH7	[31:30]	rw	**Reset CN0 of channel 7 on force update event** See bits 17:16

24.6.6.8 Register TOMi_TGC0_INT_TRIG

GTM_TOMi_TGC0_INT_TRIG (i=0-1)
TOMi TGC0 Internal Trigger Control Register
($0803C_H$+i*800_H) **Reset Value: 00000000_H**

GTM_TOMi_TGC1_INT_TRIG (i=0-1)
TOMi TGC1 Internal Trigger Control Register
($0823C_H$+i*800_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
INT_TRIG0	[1:0]	rw	**Select input signal TRIG_0 as a trigger source** Write of following double bit values is possible: 00_B don't care, bits 1:0 will not be changed 01_B internal trigger from channel 0 (TRIG_0) not used: is read as 00 (see below) 10_B internal trigger from channel 0 (TRIG_0) used: is read as 11 (see below) 11_B don't care, bits 1:0 will not be changed Read of following double values means: 00_B internal trigger from channel 0 (TRIG_0) not used 11_B internal trigger from channel 0 (TRIG_0) used
INT_TRIG1	[3:2]	rw	**Select input signal TRIG_1 as a trigger source** See bits 1:0
INT_TRIG2	[5:4]	rw	**Select input signal TRIG_2 as a trigger source** See bits 1:0
INT_TRIG3	[7:6]	rw	**Select input signal TRIG_3 as a trigger source** See bits 1:0
INT_TRIG4	[9:8]	rw	**Select input signal TRIG_4 as a trigger source** See bits 1:0

Field	Bits	Type	Description
INT_TRIG5	[11:10]	rw	**Select input signal TRIG_5 as a trigger source** See bits 1:0
INT_TRIG6	[13:12]	rw	**Select input signal TRIG_6 as a trigger source** See bits 1:0
INT_TRIG7	[15:14]	rw	**Select input signal TRIG_7 as a trigger source** See bits 1:0
Reserved	[31:16]	r	**Reserved** Read as zero, should be written as zero

24.6.6.9 Register TOMi_CHx_CTRL (x:0…14)

GTM_TOM0_CHx_CTRL (x=0-14)
TOM0 Channel x Control Register'

(08000_H+x*0040_H) **Reset Value: 00000800_H**

GTM_TOM1_CHx_CTRL (x=0-14)
TOM1 Channel x Control Register'

(08800_H+x*0040_H) **Reset Value: 00000800_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		**0**	**0**	**Reserved**	**OSM**	**Reserved**	**TRIG OUT**	**EXT TRIG OUT**	**EXT_TRIG**	**OSM_TRIG**	**RST_CCU0**	**Reserved**			
r		rw	rw	r	rw	r	rw	rw	rw	rw	rw	r			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	**CLK_SRC_SR**			**SL**	**Reserved**										
r	rw			rw	r										

Field	Bits	Type	Description
Reserved	[10:0]	r	**Reserved** Read as zero, should be written as zero
SL	11	rw	**Signal level for duty cycle** 0_B Low signal level 1_B High signal level If the output is disabled, the output TOM_OUT[x] is set to inverse value of SL.

Field	Bits	Type	Description
CLK_SRC_SR	[14:12]	rw	**Clock source select for channel** The register CLK_SRC is updated with the value of CLK_SRC_SR together with the update of register CM0 and CM1. The input of the FX clock divider depends on the value of FXCLK_SEL (see CMU). 000_B CMU_FXCLK(0) selected: clock selected by FXCLKSEL 001_B CMU_FXCLK(1) selected: clock selected by FXCLKSEL/ 2^4 010_B CMU_FXCLK(2) selected: clock selected by FXCLKSEL / 2^8 011_B CMU_FXCLK(3) selected: clock selected by FXCLKSEL / 2^12 100_B CMU_FXCLK(4) selected: clock selected by FXCLKSEL / 2^16 101_B no CMU_FXCLK selected, clock of channel stopped 110_B no CMU_FXCLK selected, clock of channel stopped 111_B no CMU_FXCLK selected, clock of channel stopped Note: if clock of channel is stopped (i.e. CLK_SRC = 101/110/111), the channel can only be restarted by resetting CLK_SRC_SR to a value of 000 to 100 and forcing an update via the force update mechanism.
Reserved	[19:15]	r	**Reserved** Read as zero, should be written as zero
RST_CCU0	20	rw	**Reset source of CCU0** 0_B Reset counter register CN0 to 0 on matching comparison CM0 1_B Reset counter register CN0 to 0 on trigger TRIG_[x-1] or TIM_EXT_CAPTURE(x) *Note: This bit should only be set if bit OSM=0 (i.e. in continuos mode).*

Field	Bits	Type	Description
OSM_TRIG	21	rw	**Enable Trigger of one-Shot Pulse by trigger signal OSM_TRIG** 0_B singal OSM_TRIG can not tigger start of single pulse generation 1_B singal OSM_TRIG can tigger start of single pulse generation (only it bit OSM=1) *Note: This bit should only be set if bit OSM=1 and RST_CCU0=0.*
EXT_TRIG	22	rw	**Select TIM_EXT_CAPTURE(x) as Trigger Signal** 0_B singal TIM_[x-1] is selected as trigger to reset CN0 or to start single pulse generation 1_B singal TIM_EXT_CAPTURE(x) is selected
EXTTRIGOUT	23	rw	**Select TIM_EXT_CAPTURE(x) as Potential OUTPUT Signal TRIG_[x]** 0_B singal TRIG_[x-1] is selected as output on TRIG_[x] (if TRIGOUT=1) 1_B singal TIM_EXT_CAPTURE(x) is selected as output on TRIG_[x] (if TRIGOUT=1)
TRIGOUT	24	rw	**Trigger output selection (output signal TRIG_[x]) of module TOM_CH[x]** 0_B TRIG_[x] is TRIG_[x-1] ro TIM_EXT_CAPTURE(x) 1_B TRIG_[x] is TRIG_CCU0
Reserved	25	r	**Reserved** Read as zero, should be written as zero
OSM	26	rw	**One-shot mode** In this mode the counter CN0 counts for only one period The length of period is defined by CM0 A write access to the register CN0 triggers the start of counting 0_B One-shot mode disabled 1_B One-shot mode enabled
Reserved	27	r	**Reserved** Read as zero, should be written as zero
0	28	rw	**Reserved** Should be written as zero
0	29	rw	**Reserved** Should be written as zero

Field	Bits	Type	Description
Reserved	[31:30]	r	**Reserved** Read as zero, should be written as zero

24.6.6.10 Register TOMi_CH15_CTRL

GTM_TOMi_CH15_CTRL (i=0-1)
TOMi Channel 15 Control Register($083C0_H$+i*800_H) **Reset Value: 00000800_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				BITREV	OSM	Reserved	TRIGOUT	EXTTRIGOUT	EXT_TRIG	OSM_TRIG	RST_CCU0	Reserved			
r				rw	rw	r	rw	rw	rw	rw	rw	r			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	CLK_SRC_SR			SL	Reserved										
r	rw			rw	r										

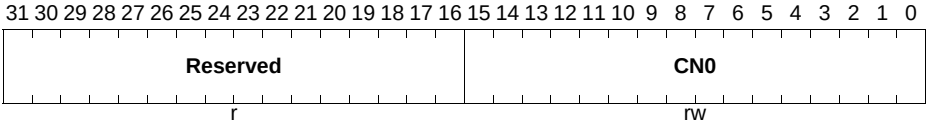
Field	Bits	Type	Description
Reserved	[10:0]	r	**Reserved** Read as zero, should be written as zero
SL	11	rw	**Signal level for duty cycle** 0_B Low signal level 1_B High signal level If the output is disabled, the output TOM_OUT[x] is set to inverse value of SL.

Field	Bits	Type	Description
CLK_SRC_SR	[14:12]	rw	**Clock source select for channel** The register CLK_SRC is updated with the value of CLK_SRC_SR together with the update of register CM0 and CM1. The input of the FX clock divider depends on the value of FXCLK_SEL (see CMU). 000_B CMU_FXCLK(0) selected: clock selected by FXCLKSEL 001_B CMU_FXCLK(1) selected: clock selected by FXCLKSEL/ 2^4 010_B CMU_FXCLK(2) selected: clock selected by FXCLKSEL / 2^8 011_B CMU_FXCLK(3) selected: clock selected by FXCLKSEL / 2^12 100_B CMU_FXCLK(4) selected: clock selected by FXCLKSEL / 2^16 101_B no CMU_FXCLK selected, clock of channel stopped 110_B no CMU_FXCLK selected, clock of channel stopped 111_B no CMU_FXCLK selected, clock of channel stopped Note: if clock of channel is stopped (i.e. CLK_SRC = 101/110/111), the channel can only be restarted by resetting CLK_SRC_SR to a value of 000 to 100 and forcing an update via the force update mechanism.
Reserved	[19:15]	r	**Reserved** Read as zero, should be written as zero
RST_CCU0	20	rw	**Reset source of CCU0** 0_B Reset counter register CN0 to 0 on matching comparison CM0 1_B Reset counter register CN0 to 0 on trigger TRIG_14

Field	Bits	Type	Description
OSM_TRIG	21	rw	**Enable Trigger of one-Shot Pulse by trigger signal OSM_TRIG** 0_B singal OSM_TRIG can not tigger start of single pulse generation 1_B singal OSM_TRIG can tigger start of single pulse generation (only it bit OSM=1) *Note: This bit should only be set if bit OSM=1 and RST_CCU0=0.*
EXT_TRIG	22	rw	**Select TIM_EXT_CAPTURE(x) as Trigger Signal** 0_B singal TIM_[x-1] is selected as trigger to reset CN0 or to start single pulse generation 1_B singal TIM_EXT_CAPTURE(x) is selected
EXTTRIGOUT	23	rw	**Select TIM_EXT_CAPTURE(x) as Potential OUTPUT Signal TRIG_[x]** 0_B singal TRIG_[x-1] is selected as output on TRIG_[x] (if TRIGOUT=1) 1_B singal TIM_EXT_CAPTURE(x) is selected as output on TRIG_[x] (if TRIGOUT=1)
TRIGOUT	24	rw	**Trigger output selection (output signal TRIG_[x]) of module TOM_CH[x]** 0_B TRIG_[x] is TRIG_[x-1] ro TIM_EXT_CAPTURE(x) 1_B TRIG_[x] is TRIG_CCU0
Reserved	25	r	**Reserved** Read as zero, should be written as zero
OSM	26	rw	**One-shot mode** In this mode the counter CN0 counts for only one period The length of period is defined by CM0 A write access to the register CN0 triggers the start of counting 0_B One-shot mode disabled 1_B One-shot mode enabled
BITREV	27	rw	**Bit-reversing of output of counter register CN0** This bit enables the PCM mode of channel 15.
Reserved	[31:28]	r	**Reserved** Read as zero, should be written as zero

24.6.6.11 Register TOMi_CHx_CN0 (x:0…15)

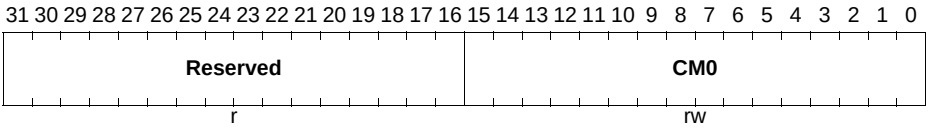
GTM_TOM0_CHx_CN0 (x=0-15)
TOM0 Channel x CCU0 Counter Register
(08014_H+x*0040_H) **Reset Value: 00000000_H**
GTM_TOM1_CHx_CN0 (x=0-15)
TOM1 Channel x CCU0 Counter Register
(08814_H+x*0040_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
CN0	[15:0]	rw	**TOM CCU0 counter register** This counter is stopped if the TOM channel is disabled and not reset on an enable event of TOM channel.
Reserved	[31:16]	r	**Reserved** Read as zero, should be written as zero

24.6.6.12 Register TOMi_CHx_CM0 (x:0…15)

GTM_TOM0_CHx_CM0 (x=0-15)
TOM0 Channel x CCU0 Compare Register
($0800C_H$+x*0040_H) **Reset Value: 00000000_H**
GTM_TOM1_CHx_CM0 (x=0-15)
TOM1 Channel x CCU0 Compare Register
($0880C_H$+x*0040_H) **Reset Value: 00000000_H**

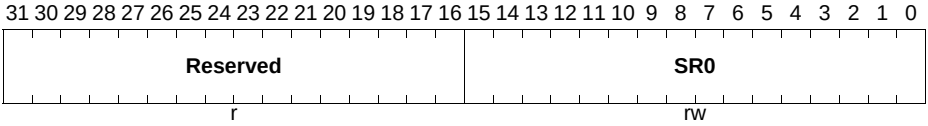


Field	Bits	Type	Description
CM0	[15:0]	rw	**TOM CCU0 compare register** Setting CM0 < CM1 configures a duty cycle of 100%.

Field	Bits	Type	Description
Reserved	[31:16]	r	**Reserved** Read as zero, should be written as zero

24.6.6.13 Register TOMi_CHx_SR0 (x:0…15)

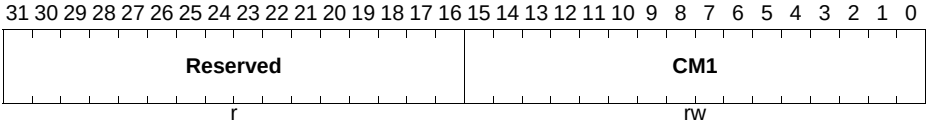
GTM_TOM0_CHx_SR0 (x=0-15)
TOM0 Channel x CCU0 Compare Shadow Register
(08004_H+x*0040_H) **Reset Value: 00000000_H**
GTM_TOM1_CHx_SR0 (x=0-15)
TOM1 Channel x CCU0 Compare Shadow Register
(08804_H+x*0040_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
SR0	[15:0]	rw	**TOM channel x shadow register SR0 for update of compare register CM0**
Reserved	[31:16]	r	**Reserved** Read as zero, should be written as zero

24.6.6.14 Register TOMi_CHx_CM1 (x:0…15)

GTM_TOM0_CHx_CM1 (x=0-15)
TOM0 Channel x CCU1 Compare Register
(08010_H+x*0040_H) **Reset Value: 00000000_H**
GTM_TOM1_CHx_CM1 (x=0-15)
TOM1 Channel x CCU1 Compare Register
(08810_H+x*0040_H) **Reset Value: 00000000_H**

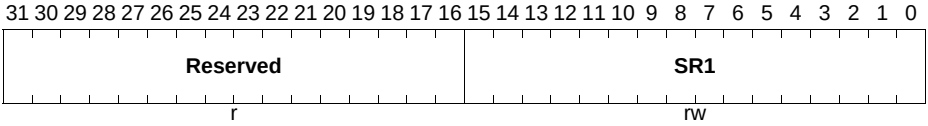


Field	Bits	Type	Description
CM1	[15:0]	rw	**TOM CCU1 compare register** Setting CM1 = 0 configures a duty cycle of 0% independent of the configured value of CM0.

Field	Bits	Type	Description
Reserved	[31:16]	r	**Reserved** Read as zero, should be written as zero

24.6.6.15 Register TOMi_CHx_SR1 (x:0…15)

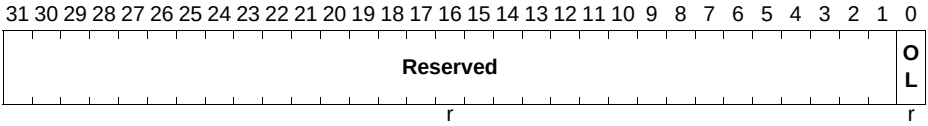
GTM_TOM0_CHx_SR1 (x=0-15)
TOM0 Channel x CCU1 Compare Shadow Register
(08008_H+x*0040_H) **Reset Value: 00000000_H**
GTM_TOM1_CHx_SR1 (x=0-15)
TOM1 Channel x CCU1 Compare Shadow Register
(08808_H+x*0040_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
SR1	[15:0]	rw	**TOM channel x shadow register SR1 for update of compare register CM1**
Reserved	[31:16]	r	**Reserved** Read as zero, should be written as zero

24.6.6.16 Register TOMi_CHx_STAT (x:0…15)

GTM_TOM0_CHx_STAT (x=0-15)
TOM0 Channel Status Register
(08018_H+x*0040_H) **Reset Value: 00000000_H**
GTM_TOM1_CHx_STAT (x=0-15)
TOM1 Channel Status Register
(08818_H+x*0040_H) **Reset Value: 00000000_H**

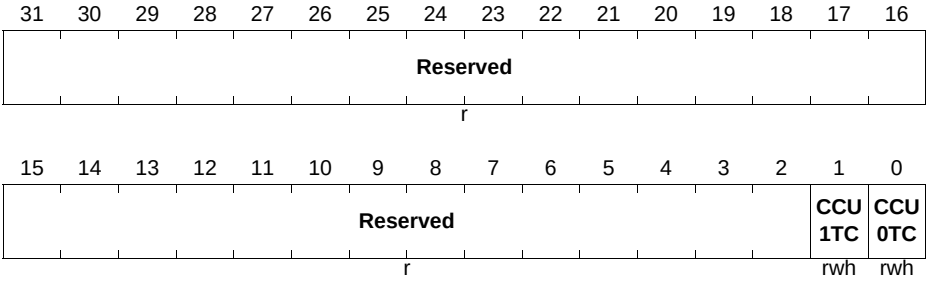


Field	Bits	Type	Description
OL	0	r	**Output level of output TOM_OUT(x)**
Reserved	[31:1]	r	**Reserved** Read as zero, should be written as zero

24.6.6.17 Register TOMi_CHx_IRQ_NOTIFY (x:0…15)

GTM_TOM0_CHx_IRQ_NOTIFY (x=0-15)
TOM0 Channel x Interrupt Notification Register
($0801C_H$+x*0040_H) **Reset Value: 00000000_H**

GTM_TOM1_CHx_IRQ_NOTIFY (x=0-15)
TOM1 Channel x Interrupt Notification Register
($0881C_H$+x*0040_H) **Reset Value: 00000000_H**

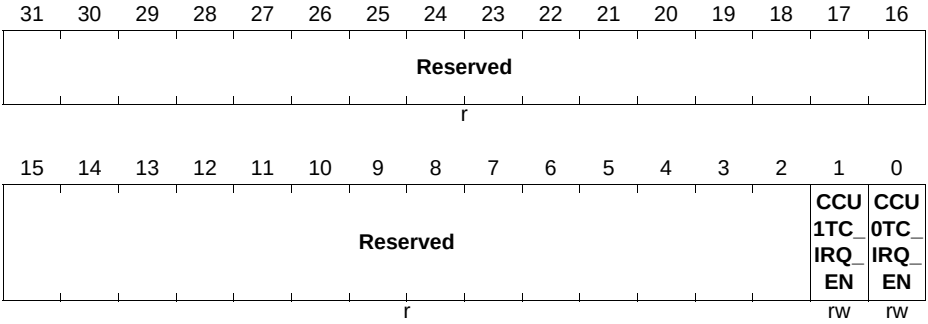


Field	Bits	Type	Description
CCU0TC	0	rwh	**CCU0 Trigger condition interrupt for channel x** 0_B No interrupt occurred 1_B The condition CN0 >= CM0 was detected. The notification of the interrupt is only triggered one time after reaching the condition CN0 >= CM0. To re-trigger the notification first the condition CN0 < CM0 has to be occurred.
CCU1TC	1	rwh	**CCU1 Trigger condition interrupt for channel x** 0_B No interrupt occurred 1_B The condition CN0 >= CM1 was detected. The notification of the interrupt is only triggered one time after reaching the condition CN0 >= CM1. To re-trigger the notification first the condition CN0 < CM1 has to be occurred.
Reserved	[31:2]	r	**Reserved** Read as zero, should be written as zero

24.6.6.18 Register TOMi_CHx_IRQ_EN (x:0…15)

GTM_TOM0_CHx_IRQ_EN (x=0-15)
TOM0 Channel x Interrupt Enable Register
(08020_H+x*0040_H) **Reset Value: 00000000_H**

GTM_TOM1_CHx_IRQ_EN (x=0-15)
TOM1 Channel x Interrupt Enable Register
(08820_H+x*0040_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
CCU0TC_IRQ_EN	0	rw	**TOM_CCU0TC_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
CCU1TC_IRQ_EN	1	rw	**TOM_CCU1TC_IRQ interrupt enable** See bit 0
Reserved	[31:2]	r	**Reserved** Read as zero, should be written as zero

24.6.6.19 Register TOMi_CHx_IRQ_FORCINT (x:0…15)

GTM_TOM0_CHx_IRQ_FORCINT (x=0-15)
TOM0 Channel x Software Interrupt Generation Register
(08024_H+x*0040_H) **Reset Value: 00000000_H**
GTM_TOM1_CHx_IRQ_FORCINT (x=0-15)
TOM1 Channel x Software Interrupt Generation Register
(08824_H+x*0040_H) **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														TRG_CCU1TC0	TRG_CCU0TC0
r														w	w

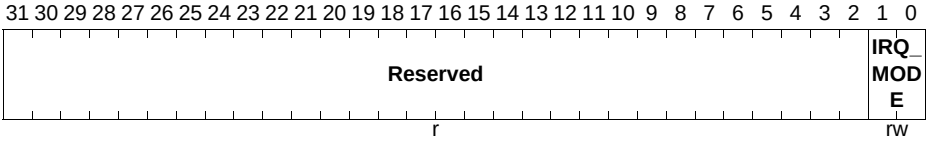
Field	Bits	Type	Description
TRG_CCU0TC0	0	w	**Trigger TOM_CCU0TC0_IRQ interrupt by software** 0_B No interrupt triggering 1_B Assert CCU0TC0_IRQ interrupt for one clock cycle Read as 0. *Note: This bit is cleared automatically after interrupt is released* Note: This bit is write protected by bit GTM_CTRL.RF_PROT.
TRG_CCU1TC0	1	w	**Trigger TOM_CCU1TC0_IRQ interrupt by software** 0_B No interrupt triggering 1_B Assert CCU1TC0_IRQ interrupt for one clock cycle Read as 0. *Note:* This bit is cleared automatically after write. Note: This bit is write protected by bit GTM_CTRL.RF_PROT.

Field	Bits	Type	Description
Reserved	[31:2]	r	**Reserved** Read as zero, should be written as zero

24.6.6.20 Register TOMi_CHx_IRQ_MODE (x:0…15)

GTM_TOM0_CHx_IRQ_MODE (x=0-15)
TOM0 IRQ Mode Configuration Register
(08028_H+x*0040_H) **Reset Value: 00000000_H**

GTM_TOM1_CHx_IRQ_MODE (x=0-15)
TOM1 IRQ Mode Configuration Register
(08828_H+x*0040_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
IRQ_MODE	[1:0]	rw	**IRQ mode selection** 00_B Level mode 01_B Pulse mode 10_B Pulse-Notify mode 11_B Single-Pulse mode Note: The interrupt modes are described in **Section 24.2.4**.
Reserved	[31:2]	r	**Reserved** Read as zero, should be written as zero

24.7 Dead Time Module

24.7.1 Overview

The following figure gives an overview of the structure of the Dead Time Module (DTM).

24.7.1.1 DTM overview

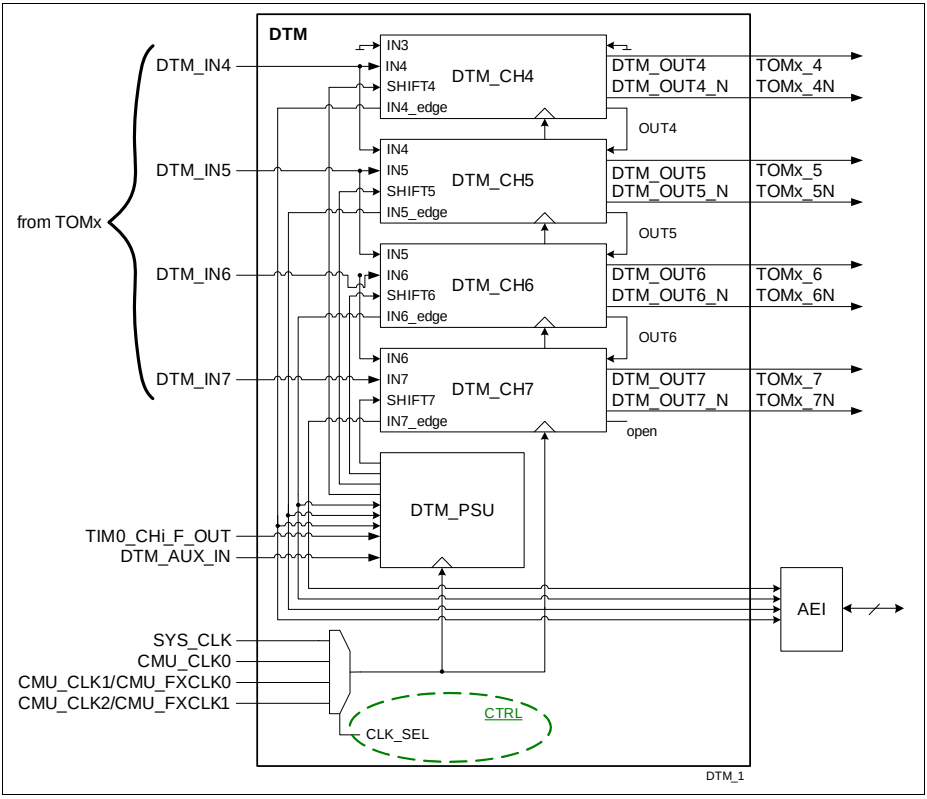


Figure 24-39 DTM Overview

The main function of the DTM is to derive for each input DTM_IN4 to DTM_IN7 the individual inverse signal (DTMi_OUT[x]_N) and to apply an edge specific delay between

the edge of the original signal and the edge of the derived inverted signal (i.e., the dead time). This function is mainly used for controlling of half bridges.

A second function provided by DTM is to set the outputs of one channel to the value of the preceding channel if requested by a trigger on input TIM0_CHi_F_OUT (TIM_CH_IN) or DTM_AUX_IN. This feature allows a phase shift on one PWM signal to the phase of the preceding PWM signal up to the next edge on this channel.

The third function provided by DTM is to (N)AND/(N)OR/X(N)OR combine the input DTM_IN[x] signal of one DTM channel with the signal on input TIM0_CHi_F_OUT (TIM_CH_IN) or DTM_AUX_IN (selected inside DTM_PSU and assigned to one of the signals SHIFT[x]) or with the output 1 (signal OUT[x]) of preceding channel.

As a result OUT5 may be the combined signal of DTM_IN4 and TIM0_CHi_F_OUT (TIM_CH_IN) or DTM_AUX_IN and the signal DTM_IN5. For OUT7 this chain can be combined again with signal DTM_IN7.

The outputs of each channel may be swapped individually to provide the function of combining signals on each output of a channel.

In general, the DTM instances are placed behind the TOM instances, i.e., the outputs TOM_OUT[x] are each routed to a DTM instance.

Additionally, some TIM instances are also connected to the DTM instances.

These connections between DTM and the modules TIM and TOM are depicted in **Section 24.7.1.2**.

24.7.1.2 Connections of TIM and TOM and to DTM

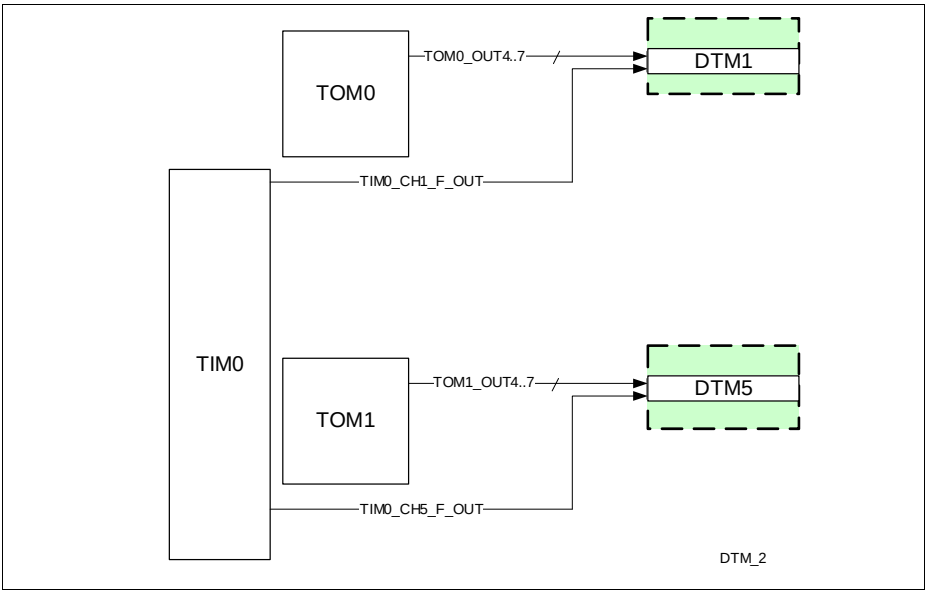


Figure 24-40 DTM Connections

For each TIM instance it can be calculated which instance of DTM and thus also which DTM channel is connected to which TIM channel or TOM channel.

24.7.2 DTM Channel

The following figure depicts the functions of a DTM channel.

24.7.2.1 DTM channel overview

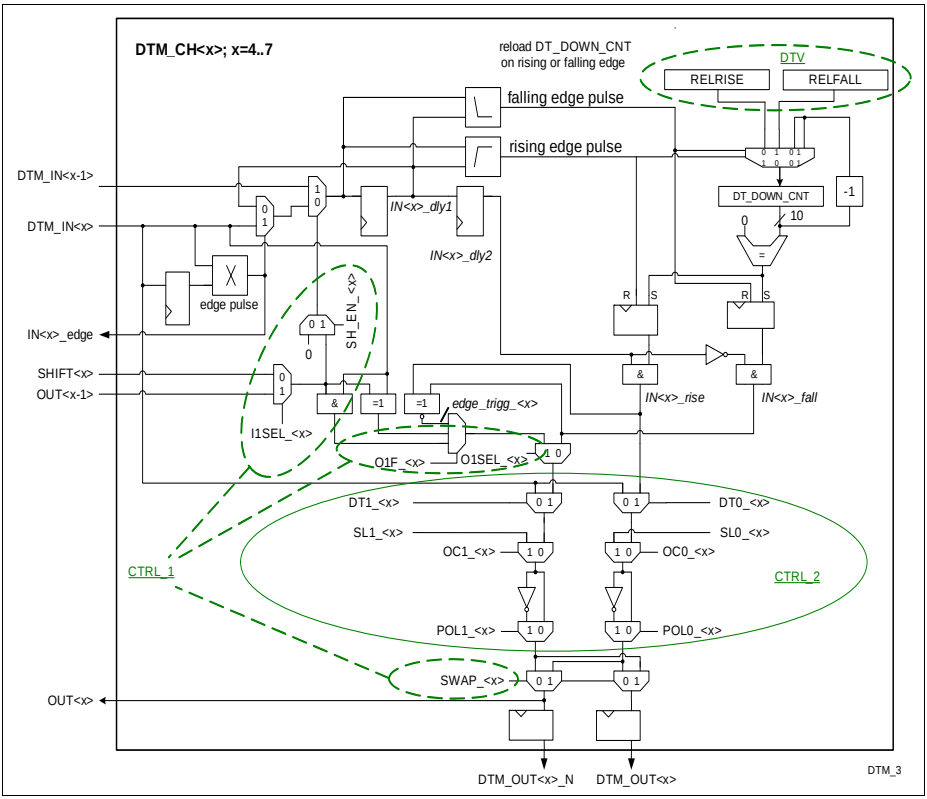


Figure 24-41 DTM Channel Overview

The main feature of each channel is to derive the inverse signal out of the input signal DTM_IN[x], apply an edge dependent delay on the two resulting signal paths and provide these signals at the outputs DTMi_OUT[x] and DTMi_OUT[x]_N.

24.7.2.2 Standard dead time generation

Standard dead time generation means that per DTM channel out of one input signal the inverse output signal is generated additionally and on both output signals the dead time between their edges is applied.

The dead time can be configured for each edge individually. The bit field RELRISE in register DTMi_CHx_DTV contains the reload value for the counter and defines the delay for rising edges in multiples of selected clock ticks.

The bit field RELFALL in register DTMi_CHx_DTV contains the reload value for the counter and defines the delay for falling edges in multiples of selected clock ticks.

The counter is reloaded with the value of RELRISE on a rising edge on input DTM_IN[x] (or DTM_IN[x-1] in case of phase shift trigger via signal SHIFT[x]) and reloaded with the value of RELFALL on a falling edge on input DTM_IN[x] (or DTM_IN[x-1] in case of shift enable SH_EN_[x].

On a reload of the counter the FlipFlop following the counter output comparator is reset and stays reset until the counter has reached 0.

After reload, the counter DT_DOWN_CNT counts down until it reaches 0 and stops at 0.

The signal flow for function of standard dead time signal generation is depicted in following figure

24.7.2.3 Wave signals for function of dead time generation

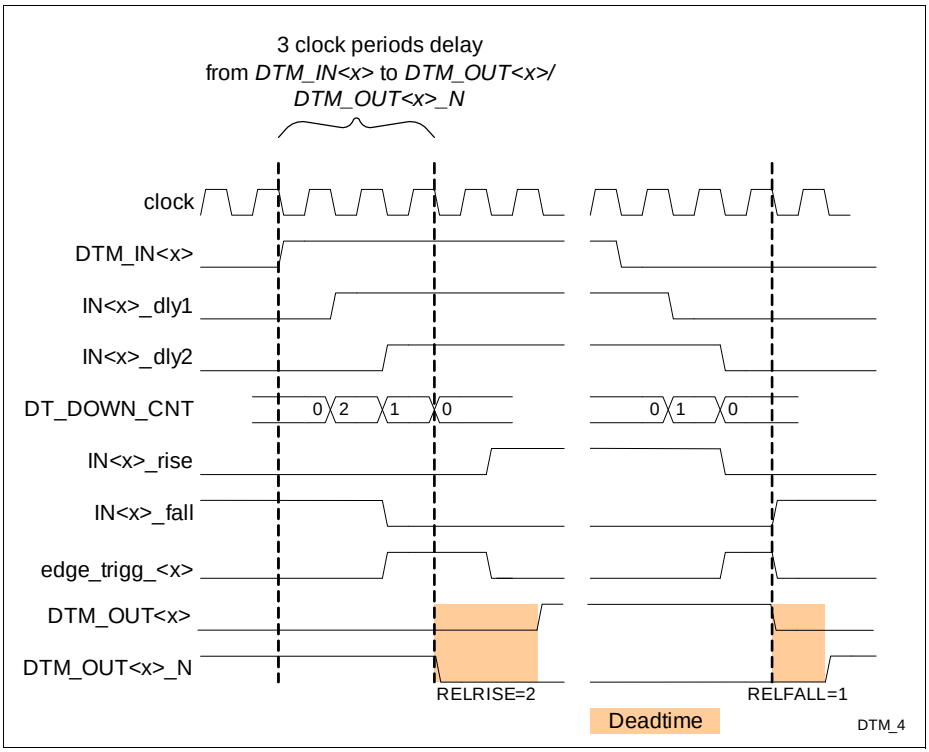


Figure 24-42 DTM Waveform

Note: The delay from the input signal DTM_IN[x] to the output signals DTMi_OUT[x] and DTMi_OUT[x]_N is three system clock periods by disabled feed through (see DT0/1_[x] in DTMi_CH_CTRL2).

Note: The delay from the input signal DTM_IN[x] to the output signals DTMi_OUT[x] and DTMi_OUT[x]_N is one system clock periods by enabled feed through (see DT0/1_[x] in DTMi_CH_CTRL2).

24.7.3 Phase Shift Control Unit

The phase shift unit (DTM_PSU) is depicted in the following figure. It supports the second major function of the DTM module to allow phase shifting of PWM signal on one of the channels.

24.7.3.1 Phase Shift Unit overview

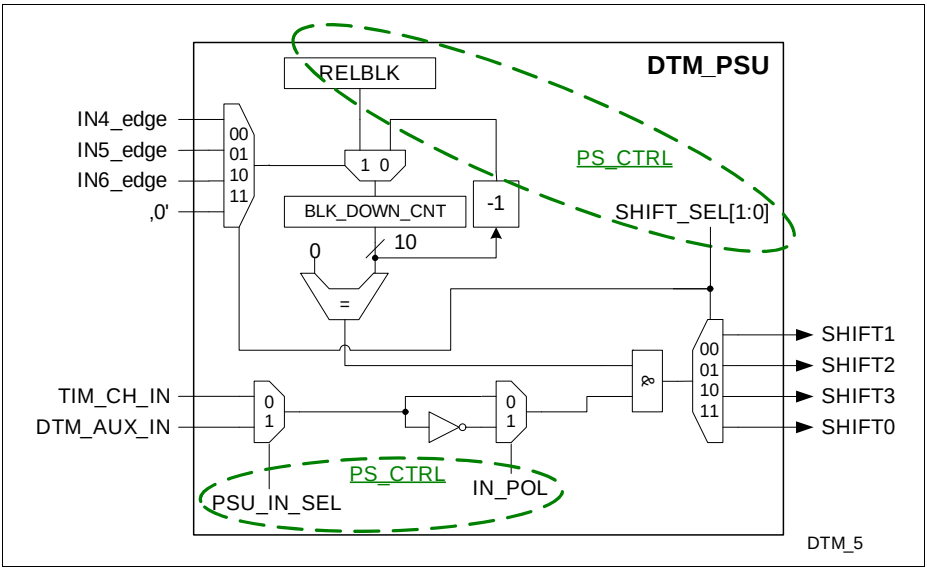


Figure 24-43 DTM Phase Shift Unit

This submodule provides an additional counter BLK_DOWN_CNT and reload bit field RELBLK (in register DTMi_PS_CTRL). The counter is reloaded on an edge detected on one of the selected signals IN0_edge to IN2_edge (selected by bit field SHIFT_SEL in register DTMi_PS_CTRL). Then, the counter counts down until it reaches 0. While the counter is counting down, it blocks the trigger (i.e. the selected one of the signals

SHIFT[x]) of one of the channels by one of the input signals TIM0_CHi_F_OUT (TIM_CH_IN) or DTM_AUX_IN.

If the counter BLK_DOWN_CNT is not counting, a pulse on the input TIM0_CHi_F_OUT (TIM_CH_IN) or DTM_AUX_IN is forwarded to one of the selected DTM_PSU outputs SHIFT[x]. This signal triggers in the selected channel (if SH_EN_x=1) the update of the first Flip-Flop on channel x (i.e. representing IN[x]_DLY) to the input value DTM_IN[x-1] of the preceding channel. If this update leads to an edge, the succeeding part of DTM channel derives the inverse signal and applies the corresponding dead time (i.e. the edge delay) to the output signals of the channel.

Note: For channel x=0 input signals DTM_IN[x-1] and OUT[x-1] are unused and I1SEL_[x] and SH_EN_[x] are defined as 0.

The following figure shows an example of phase shifting on channel 1.

24.7.3.2 Example wave of phase shift on channel 1

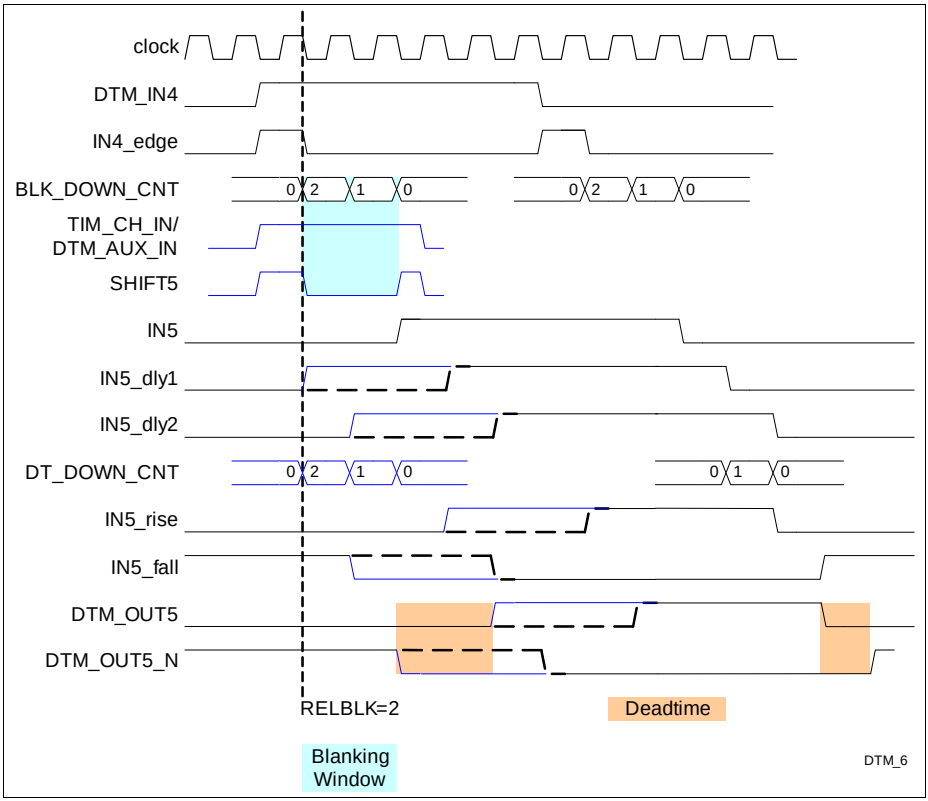


Figure 24-44 DTM Phase Shift Example

24.7.4 Multiple output signal combination

Each channel provides additionally the possibility to combine the channel inputs DTM_IN[x] and SHIFT[x] or OUT[x-1] (selected by I1SEL_[x]) by an AND or an XOR gate (selected by O1F_[x]).

It is recommended to use the combination of signals is only usable if bit field RELBLK of register DTMi_PS_CTRL is 0. Otherwise, the signals TIM0_CHi_F_OUT (TIM_CH_IN)/DTM_AUX_IN may be disturbed by the blanking window counter.

Together with the inverter inside submodule DTM_PSU (selected by IN_POL), the inverter on each output of a channel (selected by POL0_[x]/POL1_[x]) and the possibility to change polarity of DTM_IN[x] inside connected TOM channel, a (N)AND, (N)OR or X(N)OR combination of the signals is possible.

24.7.4.1 Combination of input signal TIM0_CHi_F_OUT (TIM_CH_IN)/AUX_IN with TOM signal

If the input selection I1SEL_[x] of a channel x is set to 0, the output selection O1SEL_[x] is set to 1 and SWAP_[x] is set to 0, depending on PSU_IN_SEL either TIM0_CHi_F_OUT (TIM_CH_IN) or DTM_AUX_IN can be combined with signal DTM_IN[x].

The function of combination on DTM output DTMi_OUT[x]_N (and also OUT[x]) is defined by O1F_[x] in the following way:

Table 24-20 Combination of input signal TIM0_CHi_F_OUT (TIM_CH_IN)/AUX_IN with TOM signal

	O1F_x	POL1_x	IN_POL	TOM output inverted
XOR	01	0	0	no
AND	10	0	0	no
XNOR	01	1	0	no
NAND	10	1	0	no
XNOR	01	1	1	yes
OR	10	1	1	yes
XOR	01	0	1	yes
NOR	10	0	1	yes

Note: The inversion of the TOM output can be reached by switching the SL bit (for TOM and SOMP/SOMC mode).

24.7.4.2 Combination of multiple TOM output signals

If the input selection I1SEL_[x] of a channel x (with x=1..3) is set to 1, the output selection O1SEL_[x] is set to 1 and SWAP_[x] is set to 0, the output of the preceding DTM channel OUT[x-1] can be combined with signal DTM_IN[x].

The function of combination on DTM output DTMi_OUT[x]_N (and also OUT[x]) is defined by O1F_[x] in the following way:

Table 24-21 Combination of multiple TOM output signals

	O1F_x	POL1_x	POL1_x-1	TOM output inverted
XOR	01	0	0	no
AND	10	0	0	no

Table 24-21 Combination of multiple TOM output signals (cont'd)

	O1F_x	POL1_x	POL1_x-1	TOM output inverted
XNOR	01	1	0	no
NAND	10	1	0	no
XNOR	01	1	1	yes
OR	10	1	1	yes
XOR	01	0	1	yes
NOR	10	0	1	yes

By setting I1SEL_[x] to 1 on all four channel, a combination of all four signals DTM_IN0 to DTM_IN3 can be achieved (combinatorial chain).

To allow also combination of signals generated for output DTMi_OUT[x], the outputs 0 and 1 can be swapped by setting bit SWAP_ [x] for channel x.

24.7.4.3 Pulse generation on edge

Another feature of the DTM is to generate on the second output DTM[i]_OUT[x]_N a pulse on every of corresponding input signal DTM[i]_IN[x].

This can be reached by configuring O1SEL[x] to '1', i.e. selecting signal edge_trigg[x] as the output signal (O1F[x] has to be '00'). The signal edge_trigg[x] is depicted in **Figure 24.7.2.3**.

The pulse length can be adjusted individually for each edge type by configuration value REL_RISE and REL_FALL of register DTMi_CHx_DV.

The parameter REL_RISE defines the pulse length in case of a rising edge on input DTM[i]_IN[x], the parameter REL_FALL define the pulse length in case of a falling edge in input DTM[i]_IN[x].

24.7.5 Synchronous update of channel control register 2

It is possible to use the shadow register DTMi_CH_CTRL2_SR and a selected edge of one of the channel 0 to 3 to update the work register DTMi_CH_CTRL2.

The update mechanism and its configuration is depicted in the following figure.

24.7.5.1 Synchronous update mechanism of register DTMi_CH_CTRL2

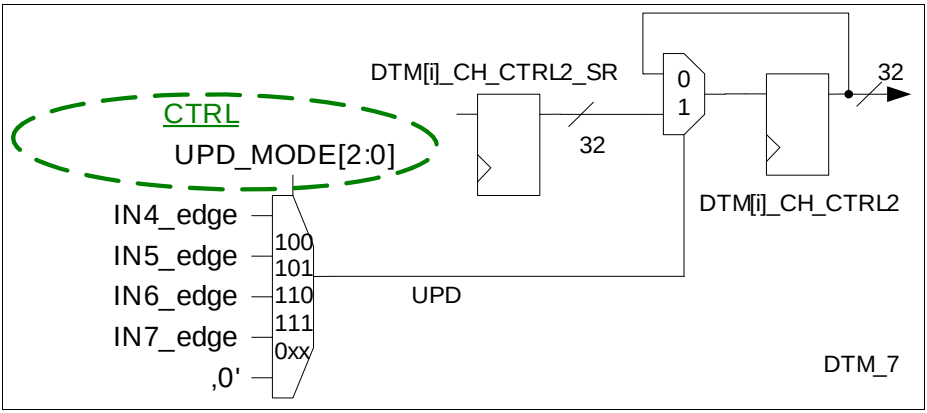


Figure 24-45 DTM Synchronous Update

If enabled by the bit field UPD_MODE of register DTMi_CTRL (i.e. UPD_MODE=1xx), the register DTMi_CH_CTRL2_SR serves as a shadow register of register DTMi_CH_CTRL2. The update is then triggered by an edge on one of the selected inputs DTM_IN4 to DTM_IN7.

The synchronous update allows the user to change output polarity, the selection of constant signal level, the constant signal level itself and the switch to/from feed through path on all four channels in parallel synchronized to one of the input edges on DTM_IN4 to DTM_IN7.

24.7.6 Configuration Register Overview

The following table gives an overview of the DTM configuration register.

Table 24-22 Configuration Register Overview

Register name	Description	Details in Section
DTMi_CTRL	Global Configuration and Control Register	13.7.1
DTMi_CH_CTRL1	Channel Control Register 1	13.7.2
DTMi_CH_CTRL2	Channel Control Register 2	13.7.3
DTMi_CH_CTRL2_SR	Channel Control Register 2 Shadow	13.7.4

Table 24-22 Configuration Register Overview (cont'd)

Register name	Description	Details in Section
DTMi_PS_CTRL	Phase Shift Unit Configuration and Control Register	13.7.5
DTMi_CHx_DTV	Dead Time Reload Values; x=0..3	13.7.6

24.7.7 Configuration Register Description

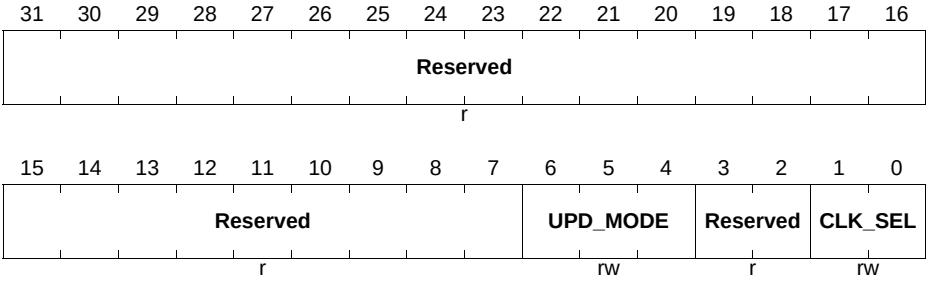
24.7.7.1 DTMi_CTRL

GTM_DTM1_CTRL
DTM1 Control Register

(13040_H) **Reset Value: 00000000_H**

GTM_DTM5_CTRL
DTM5 Control Register

(13140_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
CLK_SEL	[1:0]	rw	**Clock source select** 00_B SYS_CLK selected 01_B CMU_CLK0 selected 10_B FXCLK_0 selected 11_B FXCLK_1 selected
Reserved	[3:2]	r	**Reserved** Read as zero, should be written as zero

Field	Bits	Type	Description
UPD_MODE	[6:4]	rw	**update mode** 0--: asynchronous update - DTMi_CH_CTRL2_SR not used for update of DTMi_CH_CTRL2 100_B Signal IN0_edge used to trigger update of DTMi_CH_CTRL2 with content of DTMi_CH_CTRL2_SR 101_B Signal IN1_edge used to trigger update of DTMi_CH_CTRL2 with content of DTMi_CH_CTRL2_SR 110_B Signal IN2_edge used to trigger update of DTMi_CH_CTRL2 with content of DTMi_CH_CTRL2_SR 111_B Signal IN3_edge used to trigger update of DTMi_CH_CTRL2 with content of DTMi_CH_CTRL2_SR
Reserved	[31:7]	r	**Reserved** Read as zero, should be written as zero

24.7.7.2 DTMi_CH_CTRL1

GTM_DTM1_CH_CTRL1
DTM1 Channel Control1 Register

(13044_H) **Reset Value: 00000000_H**

GTM_DTM5_CH_CTRL1
DTM5 Channel Control1 Register

(13144_H) **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		O1F_3		SWAP_3	SH_EN_3	I1SEL_3	O1SEL_3	Reserved		O1F_2		SWAP_2	SH_EN_2	I1SEL_2	O1SEL_2
r		rw		rw	rw	rw	rw	r		rw		rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		O1F_1		SWAP_1	SH_EN_1	I1SEL_1	O1SEL_1	Reserved		O1F_0		SWAP_0	Reserved		O1SEL_0
r		rw		rw	rw	rw	rw	r		rw		rw	r		rw

Field	Bits	Type	Description
O1SEL_0	0	rw	**Output 1 select channel 0** 0_B inverse dead time signal selected 1_B special function on output 1 selected (defined by O1F_0)
Reserved	[2:1]	r	**Reserved** Read as zero, should be written as zero
SWAP_0	3	rw	**Swap outputs DTMi_CH[0]_OUT0 and DTMi_CH[0]_OUT1** (before final output register) 0_B outputs not swapped 1_B swap outputs DTMi_OUT0 and DTMi_OUT0_N
O1F_0	[5:4]	rw	**Output 1 function channel 0** 00_B Signal edge_trigg is selected 01_B XOR of DTM_IN0 and signal SHIFT0 10_B AND of DTM_IN0 and signal SHIFT0 11_B Reserved, same function as 00
Reserved	[7:6]	r	**Reserved** Read as zero, should be written as zero

Field	Bits	Type	Description
O1SEL_1	8	rw	**Output 1 select channel 1** 0_B inverse dead time signal selected 1_B special function on output 1 selected (defined by O1F_1)
I1SEL_1	9	rw	**Input 1 select channel 1** 0_B signal SHIFT1 selected 1_B signal OUT1 selected
SH_EN_1	10	rw	**Shift enable channel 1** 0_B DTM[i]_IN0 is not used; no input signal shift 1_B signal selected by I1SEL_1 triggers update of DTM[i]_IN1 with input of DTM[i]_IN0 -> input signal shift
SWAP_1	11	rw	**Swap outputs DTMi_CH[1]_OUT0 and DTMi_CH[1]_OUT1** (before final output register) 0_B outputs not swapped 1_B swap outputs DTMi_OUT1 and DTMi_OUT1_N
O1F_1	[13:12]	rw	**Output 1 function channel 1** 00_B Signal edge_trigg is selected 01_B XOR of DTM_IN1 and signal SHIFT1/OUT0 10_B AND of DTM_IN1 and signal SHIFT1/OUT0 11_B Reserved, same function as 00
Reserved	[15:14]	r	**Reserved** Read as zero, should be written as zero
O1SEL_2	16	rw	**Output 1 select channel 2** 0_B inverse dead time signal selected 1_B special function on output 1 selected (defined by O1F_2)
I1SEL_2	17	rw	**Input 1 select channel 2** 0_B signal SHIFT1 selected 1_B signal OUT1 selected
SH_EN_2	18	rw	**Shift enable channel 2** 0_B DTM[i]_IN1 is not used; no input signal shift 1_B signal selected by I1SEL_2 triggers update of DTM[i]_IN2 with input of DTM[i]_IN1 -> input signal shift

Field	Bits	Type	Description
SWAP_2	19	rw	**Swap outputs DTMi_CH[2]_OUT0 and DTMi_CH[2]_OUT1** (before final output register) 0_B outputs not swapped 1_B swap outputs DTMi_OUT2 and DTMi_OUT2_N
O1F_2	[21:20]	rw	**Output 1 function channel 2** 00_B Signal edge_trigg is selected 01_B XOR of DTM_IN2 and signal SHIFT2/OUT1 10_B AND of DTM_IN2 and signal SHIFT2/OUT1 11_B Reserved, same function as 00
Reserved	[23:22]	r	**Reserved** Read as zero, should be written as zero
O1SEL_3	24	rw	**Output 1 select channel 3** 0_B inverse dead time signal selected 1_B special function on output 1 selected (defined by O1F_3)
I1SEL_3	25	rw	**Input 1 select channel 3** 0_B signal SHIFT2 selected 1_B signal OUT2 selected
SH_EN_3	26	rw	**Shift enable channel 3** 0_B DTM[i]_IN2 is not used; no input signal shift 1_B signal selected by I1SEL_3 triggers update of DTM[i]_IN3 with input of DTM[i]_IN2 -> input signal shift
SWAP_3	27	rw	**Swap outputs DTMi_CH[3]_OUT0 and DTMi_CH[3]_OUT1** (before final output register) 0_B outputs not swapped 1_B swap outputs DTMi_OUT3 and DTMi_OUT3_N
O1F_3	[29:28]	rw	**Output 1 function channel 3** 00_B Signal edge_trigg is selected 01_B XOR of DTM_IN3 and signal SHIFT3/OUT2 10_B AND of DTM_IN3 and signal SHIFT3/OUT2 11_B Reserved, same function as 00
Reserved	[31:30]	r	**Reserved** Read as zero, should be written as zero

24.7.7.3 DTMi_CH_CTRL2

GTM_DTM1_CH_CTRL2
DTM1 Channel Control2 Register

(13048_H) **Reset Value: 00000000_H**

GTM_DTM5_CH_CTRL2
DTM5 Channel Control2 Register

(13148_H) **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DT1_3	SL1_3	OC1_3	POL1_3	DT0_3	SL0_3	OC0_3	POL0_3	DT1_2	SL1_2	OC1_2	POL1_2	DT0_2	SL0_2	OC0_2	POL0_2
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DT1_1	SL1_1	OC1_1	POL1_1	DT0_1	SL0_1	OC0_1	POL0_1	DT1_0	SL1_0	OC1_0	POL1_0	DT0_0	SL0_0	OC0_0	POL0_0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
POL0_0	0	rw	**Polarity on output 0 channel 0** 0_B Output signal not inverted 1_B Output signal inverted
OC0_0	1	rw	**Output 0 control channel 0** 0_B Functional output 1_B Constant output defined by SL0_0
SL0_0	2	rw	**Signal level on output 0 channel 0** 0_B Signal Level is '0' on output if OC0_0=1 1_B Signal Level is '1' on output if OC0_0=1
DT0_0	3	rw	**Dead time path enable on output 0 channel 0** 0_B feed through from DTM_IN0 to DTMi_OUT0 enabled 1_B dead time path enabled
POL1_0	4	rw	**Polarity on output 1 channel 0** 0_B Output signal not inverted 1_B Output signal inverted
OC1_0	5	rw	**Output 1 control channel 0** 0_B Functional output 1_B Constant output defined by SL1_0

Field	Bits	Type	Description
SL1_0	6	rw	**Signal level on output 1 channel 0** 0_B Signal Level is '0' on output if OC1_0=1 1_B Signal Level is '1' on output if OC1_0=1
DT1_0	7	rw	**Dead time path enable on output 1 channel 0** 0_B feed through from DTM_IN0 to DTMi_OUT0_N enabled 1_B dead time path enabled
POL0_1	8	rw	**Polarity on output 0 channel 1** 0_B Output signal not inverted 1_B Output signal inverted
OC0_1	9	rw	**Output 0 control channel 1** 0_B Functional output 1_B Constant output defined by SL0_1
SL0_1	10	rw	**Signal level on output 0 channel 1** 0_B Signal Level is '0' on output if OC0_1=1 1_B Signal Level is '1' on output if OC0_1=1
DT0_1	11	rw	**Dead time path enable on output 0 channel 1** 0_B feed through from DTM_IN1 to DTMi_OUT1 enabled 1_B dead time path enabled
POL1_1	12	rw	**Polarity on output 1 channel 1** 0_B Output signal not inverted 1_B Output signal inverted
OC1_1	13	rw	**Output 1 control channel 1** 0_B Functional output 1_B Constant output defined by SL1_1
SL1_1	14	rw	**Signal level on output 1 channel 1** 0_B Signal Level is '0' on output if OC1_1=1 1_B Signal Level is '1' on output if OC1_1=1
DT1_1	15	rw	**Dead time path enable on output 1 channel 1** 0_B feed through from DTM_IN1 to DTMi_OUT1_N enabled 1_B dead time path enabled
POL0_2	16	rw	**Polarity on output 0 channel 2** 0_B Output signal not inverted 1_B Output signal inverted

Field	Bits	Type	Description
OC0_2	17	rw	**Output 0 control channel 2** 0_B Functional output 1_B Constant output defined by SL0_2
SL0_2	18	rw	**Signal level on output 0 channel 2** 0_B Signal Level is '0' on output if OC0_2=1 1_B Signal Level is '1' on output if OC0_2=1
DT0_2	19	rw	**Dead time path enable on output 0 channel 2** 0_B feed through from DTM_IN2 to DTMi_OUT2 enabled 1_B dead time path enabled
POL1_2	20	rw	**Polarity on output 1 channel 2** 0_B Output signal not inverted 1_B Output signal inverted
OC1_2	21	rw	**Output 1 control channel 2** 0_B Functional output 1_B Constant output defined by SL1_2
SL1_2	22	rw	**Signal level on output 1 channel 2** 0_B Signal Level is '0' on output if OC1_2=1 1_B Signal Level is '1' on output if OC1_2=1
DT1_2	23	rw	**Dead time path enable on output 1 channel 2** 0_B feed through from DTM_IN2 to DTMi_OUT2_N enabled 1_B dead time path enabled
POL0_3	24	rw	**Polarity on output 0 channel 3** 0_B Output signal not inverted 1_B Output signal inverted
OC0_3	25	rw	**Output 0 control channel 3** 0_B Functional output 1_B Constant output defined by SL0_3
SL0_3	26	rw	**Signal level on output 0 channel 3** 0_B Signal Level is '0' on output if OC0_3=1 1_B Signal Level is '1' on output if OC0_3=1
DT0_3	27	rw	**Dead time path enable on output 0 channel 3** 0_B feed through from DTM_IN3 to DTMi_OUT3 enabled 1_B dead time path enabled

Field	Bits	Type	Description
POL1_3	28	rw	**Polarity on output 1 channel 3** 0_B Output signal not inverted 1_B Output signal inverted
OC1_3	29	rw	**Output 1 control channel 3** 0_B Functional output 1_B Constant output defined by SL1_3
SL1_3	30	rw	**Signal level on output 1 channel 3** 0_B Signal Level is '0' on output if OC1_3=1 1_B Signal Level is '1' on output if OC1_3=1
DT1_3	31	rw	**Dead time path enable on output 1 channel 3** 0_B feed through from DTM_IN3 to DTMi_OUT3_N enabled 1_B dead time path enabled

24.7.7.4 DTMi_CH_CTRL2_SR

GTM_DTM1_CH_CTRL2_SR
DTM1 Channel Control2 Shadow Register
(1304C$_H$) **Reset Value: 00000000$_H$**

GTM_DTM5_CH_CTRL2_SR
DTM5 Channel Control2 Shadow Register
(1314C$_H$) **Reset Value: 00000000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DT1_3_SR	SL1_3_SR	OC1_3_SR	POL1_3_SR	DT0_3_SR	SL0_3_SR	OC0_3_SR	POL0_3_SR	DT1_2_SR	SL1_2_SR	OC1_2_SR	POL1_2_SR	DT0_2_SR	SL0_2_SR	OC0_2_SR	POL0_2_SR
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DT1_1_SR	SL1_1_SR	OC1_1_SR	POL1_1_SR	DT0_1_SR	SL0_1_SR	OC0_1_SR	POL0_1_SR	DT1_0_SR	SL1_0_SR	OC1_0_SR	POL1_0_SR	DT0_0_SR	SL0_0_SR	OC0_0_SR	POL0_0_SR
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
POL0_0_SR	0	rw	**Polarity on output 0 channel 0 shadow register** 0_B Output signal not inverted 1_B Output signal inverted
OC0_0_SR	1	rw	**Output 0 control channel 0 shadow register** 0_B Functional output 1_B Constant output defined by SL0_0
SL0_0_SR	2	rw	**Signal level on output 0 channel 0 shadow register** 0_B Signal Level is '0' on output if OC0_0=1 1_B Signal Level is '1' on output if OC0_0=1
DT0_0_SR	3	rw	**Dead time path enable on output 0 channel 0 shadow register** 0_B feed through from DTM_IN0 to DTMi_OUT0 enabled 1_B dead time path enabled
POL1_0_SR	4	rw	**Polarity on output 1 channel 0 shadow register** 0_B Output signal not inverted 1_B Output signal inverted

Field	Bits	Type	Description
OC1_0_SR	5	rw	**Output 1 control channel 0 shadow register** 0_B Functional output 1_B Constant output defined by SL1_0
SL1_0_SR	6	rw	**Signal level on output 1 channel 0 shadow register** 0_B Signal Level is '0' on output if OC1_0=1 1_B Signal Level is '1' on output if OC1_0=1
DT1_0_SR	7	rw	**Dead time path enable on output 1 channel 0 shadow register** 0_B feed through from DTM_IN0 to DTMi_OUT0_N enabled 1_B dead time path enabled
POL0_1_SR	8	rw	**Polarity on output 0 channel 1 shadow register** 0_B Output signal not inverted 1_B Output signal inverted
OC0_1_SR	9	rw	**Output 0 control channel 1 shadow register** 0_B Functional output 1_B Constant output defined by SL0_1
SL0_1_SR	10	rw	**Signal level on output 0 channel 1 shadow register** 0_B Signal Level is '0' on output if OC0_1=1 1_B Signal Level is '1' on output if OC0_1=1
DT0_1_SR	11	rw	**Dead time path enable on output 0 channel 1 shadow register** 0_B feed through from DTM_IN1 to DTMi_OUT1 enabled 1_B dead time path enabled
POL1_1_SR	12	rw	**Polarity on output 1 channel 1 shadow register** 0_B Output signal not inverted 1_B Output signal inverted
OC1_1_SR	13	rw	**Output 1 control channel 1 shadow register** 0_B Functional output 1_B Constant output defined by SL1_1
SL1_1_SR	14	rw	**Signal level on output 1 channel 1 shadow register** 0_B Signal Level is '0' on output if OC1_1=1 1_B Signal Level is '1' on output if OC1_1=1
DT1_1_SR	15	rw	**Dead time path enable on output 1 channel 1 shadow register** 0_B feed through from DTM_IN1 to DTMi_OUT1_N 1_B dead time path enabled

Field	Bits	Type	Description
POL0_2_SR	16	rw	**Polarity on output 0 channel 2 shadow register** 0_B Output signal not inverted 1_B Output signal inverted
OC0_2_SR	17	rw	**Output 0 control channel 2 shadow register** 0_B Functional output 1_B Constant output defined by SL0_2
SL0_2_SR	18	rw	**Signal level on output 0 channel 2 shadow register** 0_B Signal Level is '0' on output if OC0_2=1 1_B Signal Level is '1' on output if OC0_2=1
DT0_2_SR	19	rw	**Dead time path enable on output 0 channel 2 shadow register** 0_B feed through from DTM_IN2 to DTMi_OUT2 1_B dead time path enabled
POL1_2_SR	20	rw	**Polarity on output 1 channel 2 shadow register** 0_B Output signal not inverted 1_B Output signal inverted
OC1_2_SR	21	rw	**Output 1 control channel 2 shadow register** 0_B Functional output 1_B Constant output defined by SL1_2
SL1_2_SR	22	rw	**Signal level on output 1 channel 2 shadow register** 0_B Signal Level is '0' on output if OC1_2=1 1_B Signal Level is '1' on output if OC1_2=1
DT1_2_SR	23	rw	**Dead time path enable on output 1 channel 2 shadow register** 0_B feed through from DTM_IN2 to DTMi_OUT2_N 1_B dead time path enabled
POL0_3_SR	24	rw	**Polarity on output 0 channel 3 shadow register** 0_B Output signal not inverted 1_B Output signal inverted
OC0_3_SR	25	rw	**Output 0 control channel 3 shadow register** 0_B Functional output 1_B Constant output defined by SL0_3
SL0_3_SR	26	rw	**Signal level on output 0 channel 3 shadow register** 0_B Signal Level is '0' on output if OC0_3=1 1_B Signal Level is '1' on output if OC0_3=1

Field	Bits	Type	Description
DT0_3_SR	27	rw	**Dead time path enable on output 0 channel 3 shadow register** 0_B feed through from DTM_IN3 to DTMi_OUT3 1_B dead time path enabled
POL1_3_SR	28	rw	**Polarity on output 1 channel 3 shadow register** 0_B Output signal not inverted 1_B Output signal inverted
OC1_3_SR	29	rw	**Output 1 control channel 3 shadow register** 0_B Functional output 1_B Constant output defined by SL1_3
SL1_3_SR	30	rw	**Signal level on output 1 channel 3 shadow register** 0_B Signal Level is '0' on output if OC1_3=1 1_B Signal Level is '1' on output if OC1_3=1
DT1_3_SR	31	rw	**Dead time path enable on output 1 channel 3 shadow register** 0_B feed through from DTM_IN3 to DTMi_OUT3_N 1_B dead time path enabled

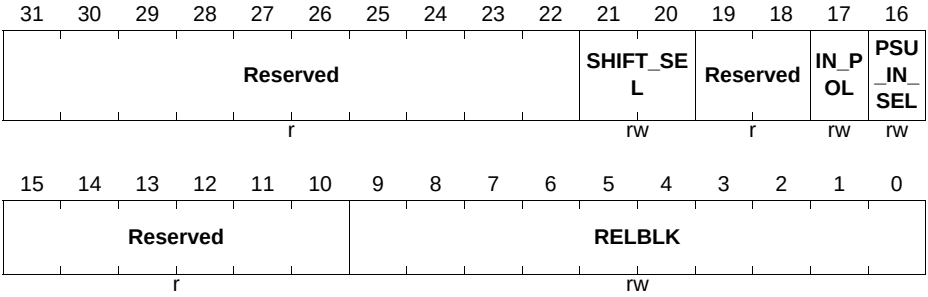
24.7.7.5 DTMi_PS_CTRL

GTM_DTM1_PS_CTRL
DTM1 Phase Shift Control Register

(13050_H) **Reset Value: 00000000_H**

GTM_DTM5_PS_CTRL
DTM5 Phase Shift Control Register

(13150_H) **Reset Value: 00000000_H**



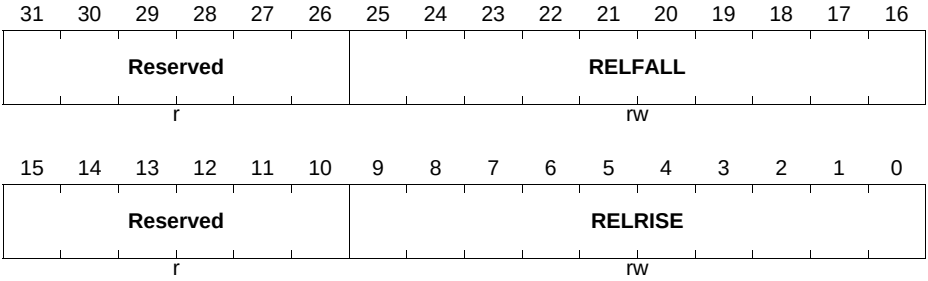
Field	Bits	Type	Description
RELBLK	[9:0]	rw	**Reload value blanking window** Note: a value of 0x000 resets counter BLK_DOWN_CNT
Reserved	[15:10]	r	**Reserved** Read as zero, should be written as zero
PSU_IN_SEL	16	rw	**PSU input select** 0_B TIM0_CHi_F_OUT (TIM_CH_IN) selected 1_B DTM_AUX_IN selected
IN_POL	17	rw	**Input polarity** 0_B input signal is not inverted 1_B input signal is inverted
Reserved	[19:18]	r	**Reserved** Read as zero, should be written as zero

Field	Bits	Type	Description
SHIFT_SEL	[21:20]	rw	**Shift select** 00_B DTM channel 1 is connected via signal SHIFT1 with TIM0_CHi_F_OUT (TIM_CH_IN)/DTM_AUX_IN 01_B DTM channel 2 is connected via signal SHIFT2 with TIM0_CHi_F_OUT (TIM_CH_IN)/DTM_AUX_IN 10_B DTM channel 3 is connected via signal SHIFT3 with TIM0_CHi_F_OUT (TIM_CH_IN)/DTM_AUX_IN 11_B DTM channel 0 is connected via signal SHIFT0 with TIM0_CHi_F_OUT (TIM_CH_IN)/DTM_AUX_IN
Reserved	[31:22]	r	**Reserved** Read as zero, should be written as zero

24.7.7.6 DTMi_CHx_DTV

GTM_DTM1_CHx_DTV (x=0-3)
DTM1 Channelx Dead Time Value Register
(13054_H+x*04_H) **Reset Value: 00000000_H**

GTM_DTM5_CHx_DTV (x=0-3)
DTM5 Channelx Dead Time Value Register
(13154_H+x*04_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
RELRISE	[9:0]	rw	**Reload value for rising edge dead time**
Reserved	[15:10]	r	**Reserved** Read as zero, should be written as zero
RELFALL	[25:16]	rw	**Reload value for falling edge dead time**
Reserved	[31:26]	r	**Reserved** Read as zero, should be written as zero

24.8 Interrupt Concentrator Module (ICM)

24.8.1 Overview

The Interrupt Concentrator Module (ICM) is used to bundle the GTM interrupt lines of the individual submodules in a reasonable manner into interrupt groups. By this bundling a smaller amount of interrupt lines is visible at the outside of the GTM.

The individual interrupts of the GTM submodules and channels have to be enabled or disabled inside the submodules and channels.

The feed through architecture of bundled interrupt lines is used for the submodules AEI, TIM, and TOM.

To determine the detailed interrupt source the microcontroller has to read the submodule/channel interrupt notification register NOTIFY and serve the channel individual interrupt.

Please note, that the interrupts are only visible inside the ICM and in consequence outside of the GTM, when the interrupt is enabled inside the submodules themselves.

24.8.2 Bundling

The GTM submodule individual interrupt sources are connected to the ICM. There, the individual interrupt lines are either feed through and signalled to the outside world or bundled a second time into groups and are then signalled to the outside world.

The ICM interrupt bundling is described in the following sections.

24.8.2.1 GTM Infrastructure Interrupt Bundling

The first interrupt group contains interrupts of the infrastructure and safety components of the GTM. This interrupt group includes therefore interrupt lines coming from the AEI submodule. In this interrupt group each individual channel of the submodules has its own interrupt line to the outside world.

Thus, the active interrupt line can be used by the CPU to determine the GTM submodule channel that raised the interrupt. The interrupts are also represented in the ICM_IRQG_0 register. This register is typically not read by the CPU, but it is readable.

24.8.2.2 TIM Interrupt Bundling

Inside this group submodules which handle GTM input signals are treated. This is the case for the TIM0 submodule. Each TIM submodule channel is able to generate six (6) individual interrupts if enabled inside the TIM channel. This six interrupts are bundled into one interrupt per TIM channel connected to the ICM.

The ICM does no further bundling. Thus, for the GTM 8 interrupt lines TIM0_IRQ[y] are provided for the external microcontroller. The channel responsible for the interrupt can be determined by the raised interrupt line.

In addition, the ICM_IRQG_2 register a mirror for the TIM submodule channel interrupts and typically not read out by the CPU, but it is readable.

24.8.2.3 TOM Interrupt Bundling

For the TOM submodules, the interrupts are bundled within the ICM submodule a second time to reduce external interrupt lines. The interrupts are OR-ed in a manner that one GTM external interrupt line represents two adjacent TOM channel interrupts. For TOM[i] the bundling is shown in **TOM interrupt bundling within ICM**.

TOM interrupt bundling within ICM

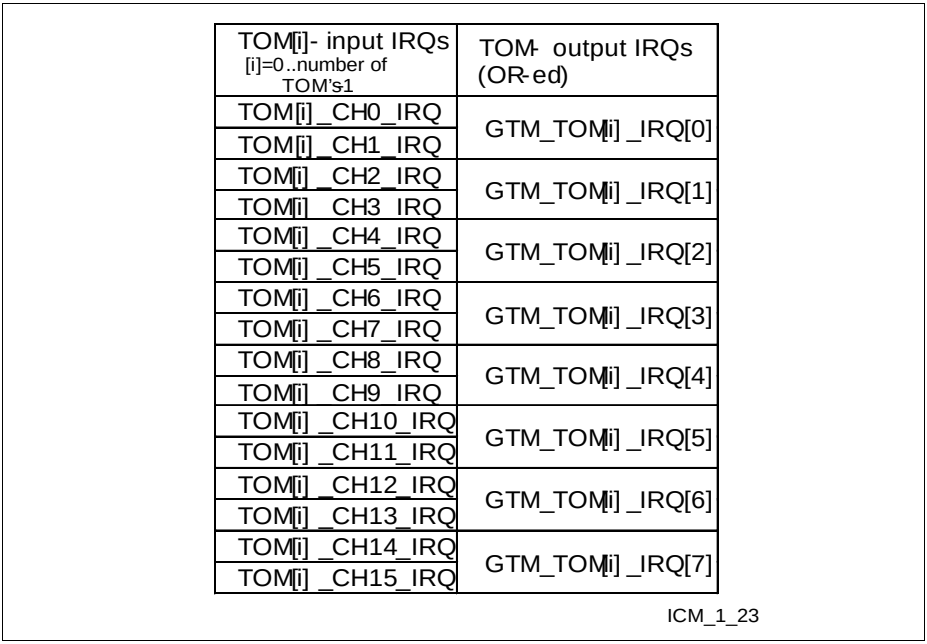


Figure 24-46 TOM interrupts

The interrupts coming from the TOM[i] submodules are registered in the ICM_IRQG_6 register. To identify the TOM submodule channel where the interrupt occurred, the CPU has to read out the ICM_IRQG_6 register first before it goes to the TOM submodule channel itself.

The ICM_IRQG_6 register bits are cleared automatically, when their corresponding interrupt in the submodule channels is cleared.

Module Error Interrupt Bundling

The Module Error Interrupt group handles the error interrupts coming from the TIM submodule of the GTM. The Module Error interrupts are additionally identified in the ICM_IRQ_MEI error interrupt group register. This register is typically not read out by the CPU, but it is readable.

The ICM_IRQG_MEI register bits are cleared automatically, when their corresponding error interrupt in the submodule is cleared.

TIM Channel Error Interrupt Bundling

The TIM Channel Error Interrupt group handles the error interrupts coming from the TIM channel of the GTM. The TIM Channel Error interrupts are additionally identified for the submodule TIM0 in the ICM_IRQ_CEI1 error interrupt group register. This register is typically not read out by the CPU, but it is readable.

The ICM_IRQG_CEI1 register bits are cleared automatically, when their corresponding error interrupt in the submodule channel is cleared.

24.8.3 ICM Interrupt Signals

Following table shows the GTM interrupt lines that are visible at the outside of the GTM.

Table 24-23 ICM Interrupt Signals

Signal	Description
GTM_AEI_IRQ	AEI Shared interrupt
GTM_TIM0_IRQ[x]	TIM Shared interrupts (x: 0…7)
GTM_TOM[i]_IRQ[x]	TOM Shared interrupts for x:0…7 = {ch0||ch1,,ch14||ch15} (i: 0…number of TOM's-1)
GTM_ERR_IRQ	GTM Error Interrupt

24.8.4 ICM Configuration Registers Overview

ICM contains following configuration registers:

Table 24-24 ICM Configuration Registers Overview

Register Name	Description	Details in Section
ICM_IRQG_0	ICM Interrupt group register covering infrastructural and safety components (AEI)	Section 24.8.5.1
ICM_IRQG_2	ICM Interrupt group register covering TIM0	Section 24.8.5.2
ICM_IRQG_6	ICM Interrupt group register covering GTM output submodules TOM0 and TOM1	Section 24.8.5.3
ICM_IRQG_MEI	ICM Interrupt group register for module error interrupt information	Section 24.8.5.4
ICM_IRQG_CEI1	ICM Interrupt group register 1 for channel error interrupt information	Section 24.8.5.5

24.8.5 ICM Configuration Registers Description

All of the following registers are 32-bit only accessible.

24.8.5.1 Register ICM_IRQG_0

GTM_ICM_IRQG_0
GTM Infrastructure Interrupt Group (600_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
Reserved	[3:0]	r	**Reserved** Read as zero, should be written as zero
AEI_IRQ	4	rh	**AEI_IRQ interrupt** See bit 0.
Reserved	[31:5]	r	**Reserved** Read as zero, should be written as zero

24.8.5.2 Register ICM_IRQG_2

GTM_ICM_IRQG_2
TIM Interrupt Group 0 **(608_H)** **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TIM0_CH7_IRQ	TIM0_CH6_IRQ	TIM0_CH5_IRQ	TIM0_CH4_IRQ	TIM0_CH3_IRQ	TIM0_CH2_IRQ	TIM0_CH1_IRQ	TIM0_CH0_IRQ
r								rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
TIM0_CH0_IRQ	0	rh	**TIM0 shared interrupt channel 0** 0_B no interrupt occurred 1_B interrupt was raised by the corresponding submodule Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule. Note: When set this bit represents one of the six interrupt sources NEWVALx_IRQ, ECNTOFLx_IRQ, CNTOFLx_IRQ, GPRzOFLx_IRQ, GLITCHDETx_IRQ or TOx_IRQ. The bit is cleared automatically, when the corresponding interrupt in the submodule channels is cleared.
TIM0_CH1_IRQ	1	rh	**TIM0 shared interrupt channel 1** See bit 0
TIM0_CH2_IRQ	2	rh	**TIM0 shared interrupt channel 2** See bit 0
TIM0_CH3_IRQ	3	rh	**TIM0 shared interrupt channel 3** See bit 0
TIM0_CH4_IRQ	4	rh	**TIM0 shared interrupt channel 4** See bit 0

Field	Bits	Type	Description
TIM0_CH5_IRQ	5	rh	**TIM0 shared interrupt channel 5** See bit 0
TIM0_CH6_IRQ	6	rh	**TIM0 shared interrupt channel 6** See bit 0
TIM0_CH7_IRQ	7	rh	**TIM0 shared interrupt channel 7** See bit 0
Reserved	[31:8]	r	**Reserved** Read as zero, should be written as zero

24.8.5.3 Register ICM_IRQG_6

GTM_ICM_IRQG_6
TOM Interrupt Group 0 **(618_H)** **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TOM1_CH15_IRQ	TOM1_CH14_IRQ	TOM1_CH13_IRQ	TOM1_CH12_IRQ	TOM1_CH11_IRQ	TOM1_CH10_IRQ	TOM1_CH9_IRQ	TOM1_CH8_IRQ	TOM1_CH7_IRQ	TOM1_CH6_IRQ	TOM1_CH5_IRQ	TOM1_CH4_IRQ	TOM1_CH3_IRQ	TOM1_CH2_IRQ	TOM1_CH1_IRQ	TOM1_CH0_IRQ
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOM0_CH15_IRQ	TOM0_CH14_IRQ	TOM0_CH13_IRQ	TOM0_CH12_IRQ	TOM0_CH11_IRQ	TOM0_CH10_IRQ	TOM0_CH9_IRQ	TOM0_CH8_IRQ	TOM0_CH7_IRQ	TOM0_CH6_IRQ	TOM0_CH5_IRQ	TOM0_CH4_IRQ	TOM0_CH3_IRQ	TOM0_CH2_IRQ	TOM0_CH1_IRQ	TOM0_CH0_IRQ
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

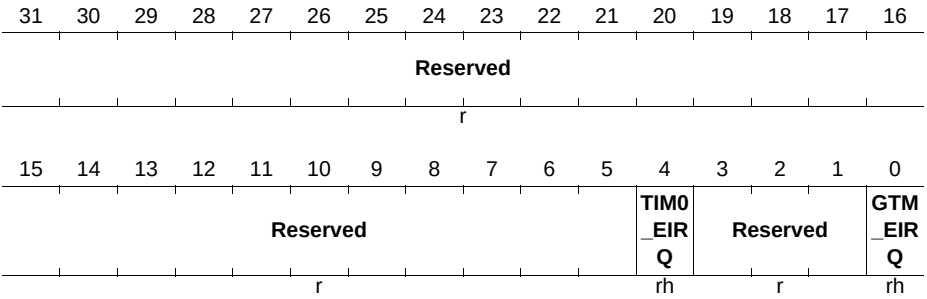
Field	Bits	Type	Description
TOM0_CH0_IRQ	0	rh	**TOM0 channel 0 shared interrupt** 0_B no interrupt occurred 1_B interrupt was raised by the corresponding submodule Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule. The bit is cleared automatically, when the corresponding interrupt in the submodule channels is cleared.
TOM0_CH1_IRQ	1	rh	**TOM0 channel 1 shared interrupt** See bit 0
TOM0_CH2_IRQ	2	rh	**TOM0 channel 2 shared interrupt** See bit 0
TOM0_CH3_IRQ	3	rh	**TOM0 channel 3 shared interrupt** See bit 0
TOM0_CH4_IRQ	4	rh	**TOM0 channel 4 shared interrupt** See bit 0
TOM0_CH5_IRQ	5	rh	**TOM0 channel 5 shared interrupt** See bit 0

Field	Bits	Type	Description
TOM0_CH6_IRQ	6	rh	**TOM0 channel 6 shared interrupt** See bit 0
TOM0_CH7_IRQ	7	rh	**TOM0 channel 7 shared interrupt** See bit 0
TOM0_CH8_IRQ	8	rh	**TOM0 channel 8 shared interrupt** See bit 0
TOM0_CH9_IRQ	9	rh	**TOM0 channel 9 shared interrupt** See bit 0
TOM0_CH10_IRQ	10	rh	**TOM0 channel 10 shared interrupt** See bit 0
TOM0_CH11_IRQ	11	rh	**TOM0 channel 11 shared interrupt** See bit 0
TOM0_CH12_IRQ	12	rh	**TOM0 channel 12 shared interrupt** See bit 0
TOM0_CH13_IRQ	13	rh	**TOM0 channel 13 shared interrupt** See bit 0
TOM0_CH14_IRQ	14	rh	**TOM0 channel 14 shared interrupt** See bit 0
TOM0_CH15_IRQ	15	rh	**TOM0 channel 15 shared interrupt** See bit 0
TOM1_CH0_IRQ	16	rh	**TOM1 channel 0 shared interrupt** See bit 0
TOM1_CH1_IRQ	17	rh	**TOM1 channel 1 shared interrupt** See bit 0
TOM1_CH2_IRQ	18	rh	**TOM1 channel 2 shared interrupt** See bit 0
TOM1_CH3_IRQ	19	rh	**TOM1 channel 3 shared interrupt** See bit 0
TOM1_CH4_IRQ	20	rh	**TOM1 channel 4 shared interrupt** See bit 0
TOM1_CH5_IRQ	21	rh	**TOM1 channel 5 shared interrupt** See bit 0
TOM1_CH6_IRQ	22	rh	**TOM1 channel 6 shared interrupt** See bit 0

Field	Bits	Type	Description
TOM1_CH7_IRQ	23	rh	**TOM1 channel 7 shared interrupt** See bit 0
TOM1_CH8_IRQ	24	rh	**TOM1 channel 8 shared interrupt** See bit 0
TOM1_CH9_IRQ	25	rh	**TOM1 channel 9 shared interrupt** See bit 0
TOM1_CH10_IRQ	26	rh	**TOM1 channel 10 shared interrupt** See bit 0
TOM1_CH11_IRQ	27	rh	**TOM1 channel 11 shared interrupt** See bit 0
TOM1_CH12_IRQ	28	rh	**TOM1 channel 12 shared interrupt** See bit 0
TOM1_CH13_IRQ	29	rh	**TOM1 channel 13 shared interrupt** See bit 0
TOM1_CH14_IRQ	30	rh	**TOM1 channel 14 shared interrupt** See bit 0
TOM1_CH15_IRQ	31	rh	**TOM1 channel 15 shared interrupt** See bit 0

24.8.5.4 Register ICM_IRQG_MEI (Module Error Interrupt)

GTM_ICM_IRQG_MEI
ICM Module Error Interrupt Register (630_H) **Reset Value: 00000000_H**



Field	Bits	Type	Description
GTM_EIRQ	0	rh	**GTM Error interrupt request** 0_B no interrupt occurred 1_B interrupt was raised by the corresponding submodule *Note: Note: This bit is only set, when the error interrupt is enabled in the error interrupt enable register of the corresponding submodule.*
Reserved	[3:1]	r	**Reserved** Read as zero, should be written as zero
TIM0_EIRQ	4	rh	**TIM0 error interrupt** See bit 0
Reserved	[31:5]	r	**Reserved** Read as zero, should be written as zero

24.8.5.5 Register ICM_IRQG_CEI1 (Channel Error Interrupt 1)

GTM_ICM_IRQG_CEI1
ICM Channel Error Interrupt 1 Register

(638_H) **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TIM0_CH7_EIRQ	TIM0_CH6_EIRQ	TIM0_CH5_EIRQ	TIM0_CH4_EIRQ	TIM0_CH3_EIRQ	TIM0_CH2_EIRQ	TIM0_CH1_EIRQ	TIM0_CH0_EIRQ
r								rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
TIM0_CH0_EIRQ	0	rh	**TIM0 channel 0 error interrupt** 0_B no error interrupt occurred 1_B error interrupt was raised by the corresponding submodule *Note: This bit is only set, when the error interrupt is enabled in the error interrupt enable register of the corresponding submodule.*
TIM0_CH1_EIRQ	1	rh	**TIM0 channel 1 error interrupt** See bit 0
TIM0_CH2_EIRQ	2	rh	**TIM0 channel 2 error interrupt** See bit 0
TIM0_CH3_EIRQ	3	rh	**TIM0 channel 3 error interrupt** See bit 0
TIM0_CH4_EIRQ	4	rh	**TIM0 channel 4 error interrupt** See bit 0
TIM0_CH5_EIRQ	5	rh	**TIM0 channel 5 error interrupt** See bit 0
TIM0_CH6_EIRQ	6	rh	**TIM0 channel 6 error interrupt** See bit 0

Field	Bits	Type	Description
TIM0_CH7_EIRQ	7	rh	**TIM0 channel 7 error interrupt** See bit 0
Reserved	[31:8]	r	**Reserved** Read as zero, should be written as zero

24.9 GTM Implementation

This chapter describes product specific implementation of the Generic Timer Module (GTM).

Note: After a module reset the outputs of the TOMs are high (see TOMi_CHx_CRTL.SL) and could cause unexpected triggers in other modules.

24.9.1 GTM Registers

Table 24-25 Registers Address Space

Module	Base Address	End Address	Note
GTM	$F010\ 0000_H$	$F019\ FFFF_H$	

Table 24-26 Registers Overview - GTM Control Registers

Short Name	Description	Offset Addr.	Access Mode		Reset	Description See
			Read	Write		
CLC	Clock Control Register	$9FD00_H$	U, SV	SV, E, P	Application	Page 24-212
TIM0INSEL	TIM0 Input Select Register	$9FD10_H$	U, SV	U, SV, P	Application	Page 24-234
TOUTSEL0	Timer Output Select 0 Register	$9FD30_H$	U, SV	U, SV, P	Application	Page 24-250
TOUTSEL1	Timer Output Select 1 Register	$9FD34_H$	U, SV	U, SV, P	Application	Page 24-250
TOUTSEL2	Timer Output Select 2 Register	$9FD38_H$	U, SV	U, SV, P	Application	Page 24-250
TOUTSEL3	Timer Output Select 3 Register	$9FD3C_H$	U, SV	U, SV, P	Application	Page 24-250
TOUTSEL4	Timer Output Select 4 Register	$9FD40_H$	U, SV	U, SV, P	Application	Page 24-250
TOUTSEL5	Timer Output Select 5 Register	$9FD44_H$	U, SV	U, SV, P	Application	Page 24-250
TOUTSEL6	Timer Output Select 6 Register	$9FD48_H$	U, SV	U, SV, P	Application	Page 24-250
TOUTSEL7	Timer Output Select 7 Register	$9FD4C_H$	U, SV	U, SV, P	Application	Page 24-250

Table 24-26 Registers Overview - GTM Control Registers

Short Name	Description	Offset Addr.	Access Mode		Reset	Description See
			Read	Write		
ADCTRIG0OUT0	ADC Trigger 0 Output 0 Register	$9FDB0_H$	U, SV	U, SV, P	Application	Page 24-252
ADCTRIG1OUT0	ADC Trigger 1 Output 0 Register	$9FDB8_H$	U, SV	U, SV, P	Application	Page 24-253
CANOUTSEL	CAN Output Select Register	$9FDA0_H$	U, SV	U, SV, P	Application	Page 24-254
OTBU0T	OCDS TBU0 Trigger Register	$9FDC4_H$	U, SV	U, SV, P	Debug	Page 24-263
OTBU1T	OCDS TBU1 Trigger Register	$9FDC8_H$	U, SV	U, SV, P	Debug	Page 24-264
OTBU2T	OCDS TBU2 Trigger Register	$9FDCC_H$	U, SV	U, SV, P	Debug	Page 24-265
OTSS	OCDS Trigger Set Select Register	$9FDD0_H$	U, SV	U, SV, P	Debug	Page 24-260
OTSC0	OCDS Trigger Set Control 0 Register	$9FDD4_H$	U, SV	U, SV, P	Debug	Page 24-261
ODA	OCDS Debug Access Register	$9FDDC_H$	U, SV	U, SV, P	Debug	Page 24-262
OCS	OCDS Control and Status Register	$9FDE8_H$	U, SV	U, SV, P	Debug	Page 24-213
KRSTCLR	Reset Status Clear Register	$9FDEC_H$	U, SV	SV, E, P	Application	Page 24-218
KRST1	Reset Control Register 1	$9FDF0_H$	U, SV	SV, E, P	Application	Page 24-218
KRST0	Reset Control Register 0	$9FDF4_H$	U, SV	SV, E, P	Application	Page 24-217
ACCEN1	Access Enable Register 1	$9FDF8_H$	U, SV	SV, SE	Application	Page 24-216
ACCEN0	Access Enable Register 0	$9FDFC_H$	U, SV	SV, SE	Application	Page 24-215

GTM Address Ordering

The GTM address area start at F010 0000_H and ends at F019 $FFFF_H$. All given address for registers are offset addresses to the GTM base address of F010 0000_H and need to be added.

Table 24-27 Address Overview

Submodule	Base address (to be added to GTM based address)
BRIDGE	0x00000030
TBU	0x00000100
CMU	0x00000300
ICM	0x00000600
TIM0	0x00001000
TOM0	0x00008000
TOM1	0x00008800
DTM1	0x00013000
DTM5	0x00013080

The clock control register is used to switch the GTM on or off and to control its input clock rate. The GTM can be disabled by setting bit DISR to 1.

CLC
Clock Control Register (9FD00$_H$) Reset Value: 0000 0003$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												EDIS		DIS S	DIS R
r												rw	r	r	rw

Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the GTM module. 0_B No disable requested 1_B Disable requested
DISS	1	r	**Module Disable Status Bit** Bit indicates the current status of the GTM module. 0_B GTM module is enabled 1_B GTM module is disabled
EDIS	3	rw	**Sleep Mode Enable Control** Used for module sleep mode control.
0	2, [31:4]	r	**Reserved** Read as 0; should be written with 0.

OCDS Control and Status Register (OCS)

The OCDS Control and Status (OCS) register is cleared by Debug Reset and by each System Reset when OCDS is disabled. It is not touched by System Reset when OCDS is enabled.

The OCS register includes the module related control bits for the ODDS Trigger Bus (OTGB).

The OCS control register bits are only effective while the system is in debug mode. While not in debug mode, OCS reset values/modes are effective.
Write access is 32 bit wide only and requires Supervisor Mode.

OCS
OCDS Control and Status (9FDE8$_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**SUSSTA**	**SUS_P**	**SUS**				**0**							
r		rh	w	rw				r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
SUS	[27:24]	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0_H Will not suspend 1_H Hard suspend. Clock is switched off immediately. In Hard suspend no registers beside the registers documented in the Implementation section of this chapter could be read or written. 2_H Soft suspend (GTM Halt Mode). In Soft suspend registers could be read or written, for details see **Chapter 24.2.5**. **others**, reserved
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	[23:0], [31:30]	r	**Reserved** Read as 0; must be written with 0.

The Access Enable Register 0 controls write access for transactions with the on chip bus master TAG ID 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000_B, EN1 -> TAG ID 000001_B, ,EN31 -> TAG ID 011111_B.

All registers and memories of the GTM are protected beside the following registers:

ACCEN0 and ACCEN1.

ACCEN0
Access Enable Register 0 **(9FDFC$_H$)** **Reset Value: FFFF FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN 31	**EN 30**	**EN 29**	**EN 28**	**EN 27**	**EN 26**	**EN 25**	**EN 24**	**EN 23**	**EN 22**	**EN 21**	**EN 20**	**EN 19**	**EN 18**	**EN 17**	**EN 16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN 15	**EN 14**	**EN 13**	**EN 12**	**EN 11**	**EN 10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n = 0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Access Enable Register (ACCEN1)

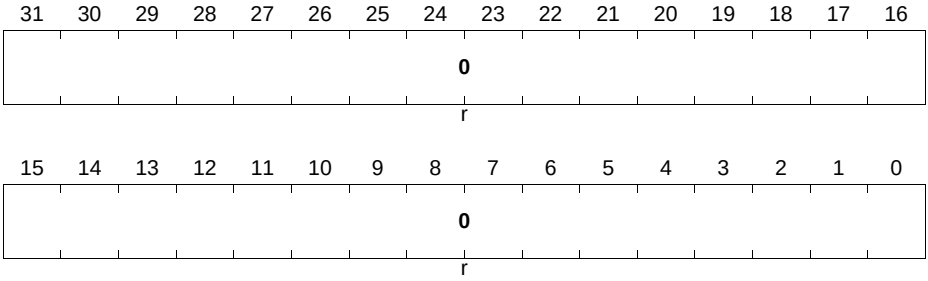
The Access Enable Register 1 controls write access for transactions with the on chip bus master TAG ID 100000_B to 111111_B (see On Chip Bus chapter for the products TAG ID master peripheral mapping).

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001_B, , EN31 -> TAG ID 111111_B.

All registers and memories of the GTM are protected beside the following registers:

ACCEN0 and ACCEN1.

ACCEN1
Access Enable Register 1 **(9FDF8$_H$)** **Reset Value: 0000 0000$_H$**



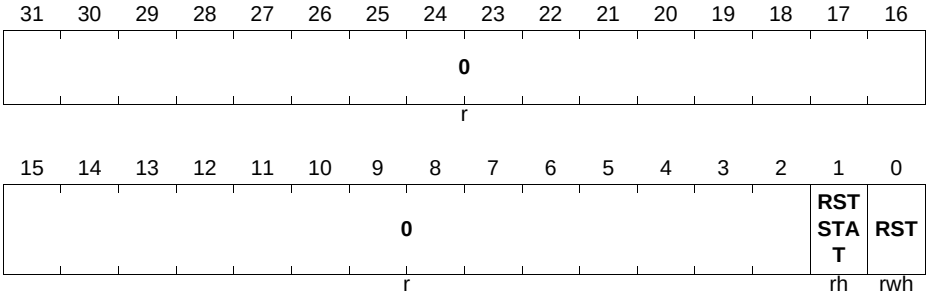
Field	Bits	Type	Description
0	[31:0]	r	**Reserved** Read as 0; should be written with 0.

The Kernel Reset Register 0 is used to reset the related module kernel. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers related to the module kernel that should be reset (kernel 0 or kernel 1). In order support modules with two kernel the BPI_FPI provides two set of kernel reset registers. The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing to it with ´1´.

Note: This reset function has the effect as bit RST.RST.

KRST0
Kernel Reset Register 0 **(9FDF4$_H$)** **Reset Value: 0000 0000$_H$**

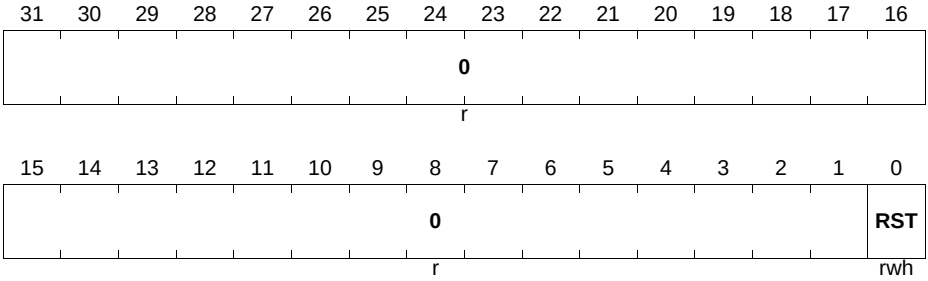


Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed.
RSTSTAT	1	rw	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. 0_B No kernel reset was executed 1_B Kernel reset was executed This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register.
0	[31:2]	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1 (KRST1)

The Kernel Reset Register 1 is used to reset the GTM kernel. GTM kernel registers related to the Debug Reset (Class 1) are not influenced. To reset the GTM kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset registers. The RST bit will be cleared with the end of the BPI kernel reset sequence.

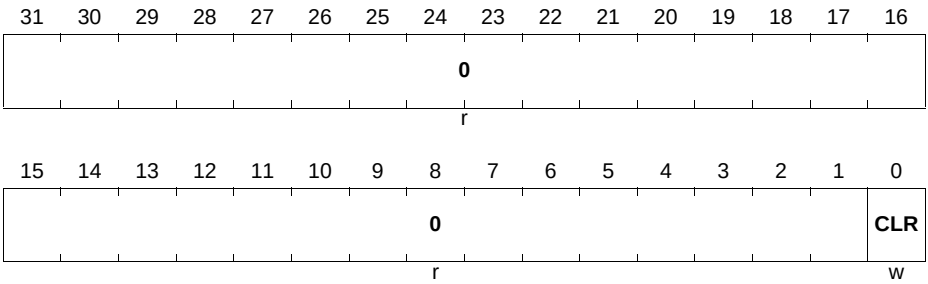
KRST1
Kernel Reset Register 1 **(9FDF0$_H$)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) after the kernel reset was executed.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

The Kernel Reset Register Clear register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR
Kernel Reset Status Clear Register (9FDEC$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT Read always as 0.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

24.9.2 Port Connections

The sections summarize the port connections which defined the boundary of the GTM for the TC21x/TC22x/TC23x.

Table 24-28 GTM to Port Mapping for QFP-80

Port	Input	Output	Input Timer Mapped		Output Timer Mapped			
			A	B	A	B	C	D
P00.0	TIN9	TOUT9	Reserved	TIM0_0	TOM0_8	TOM1_0	TOM0_4	TOM1_4
P02.0	TIN0	TOUT0	TIM0_0	Reserved	TOM0_8	TOM1_8	TOM0_4	TOM1_4
P02.1	TIN1	TOUT1	TIM0_1	Reserved	TOM0_9	TOM1_9	TOM0_4N	TOM1_4N
P02.2	TIN2	TOUT2	TIM0_2	Reserved	TOM0_10	TOM1_10	TOM0_5	TOM1_5
P02.3	TIN3	TOUT3	TIM0_3	Reserved	TOM0_11	TOM1_11	TOM0_5N	TOM1_5N
P02.4	TIN4	TOUT4	TIM0_4	Reserved	TOM0_12	TOM1_12	TOM0_6	TOM1_6
P02.5	TIN5	TOUT5	TIM0_5	Reserved	TOM0_13	TOM1_13	TOM0_6N	TOM1_6N
P02.6	TIN6	TOUT6	TIM0_6	Reserved	TOM0_14	TOM1_14	TOM0_7	TOM1_7
P02.7	TIN7	TOUT7	TIM0_7	Reserved	TOM0_15	TOM1_15	TOM0_7N	TOM1_7N
P02.8	TIN8	TOUT8	Reserved	TIM0_0	TOM0_8	TOM1_0	TOM0_4N	TOM1_4N
P10.5	TIN107	TOUT107	TIM0_2	Reserved	TOM0_2	TOM1_10	Reserved	Reserved
P10.6	TIN108	TOUT108	TIM0_3	Reserved	TOM0_3	TOM1_11	Reserved	Reserved
P11.2	Reserved	TOUT95	Reserved	Reserved	TOM0_8	TOM1_1	TOM0_4N	TOM1_4N
P11.3	Reserved	TOUT96	Reserved	Reserved	TOM0_10	TOM1_2	TOM0_5	TOM1_5
P11.6	Reserved	TOUT97	Reserved	Reserved	TOM0_11	TOM1_3	TOM0_5N	TOM1_5N

Table 24-28 GTM to Port Mapping for QFP-80

Port	Input	Output	Input Timer Mapped		Output Timer Mapped			
			A	B	A	B	C	D
P11.9	Reserved	TOUT98	Reserved	Reserved	TOM0_12	TOM1_4	TOM0_6	TOM1_6
P11.10	Reserved	TOUT99	Reserved	Reserved	TOM0_13	TOM1_5	TOM0_6N	TOM1_6N
P11.11	Reserved	TOUT100	Reserved	Reserved	TOM0_14	TOM1_6	TOM0_7N	TOM1_7N
P11.12	Reserved	TOUT101	Reserved	Reserved	TOM0_15	TOM1_7	TOM0_7	TOM1_7
P14.0	TIN80	TOUT80	TIM0_3	Reserved	TOM0_3	TOM1_3	TOM0_6	TOM1_6
P14.1	TIN81	TOUT81	TIM0_4	Reserved	TOM0_4	TOM1_4	TOM0_7	TOM1_7
P14.3	TIN83	TOUT83	TIM0_6	Reserved	TOM0_6	TOM1_6	Reserved	Reserved
P14.4	TIN84	TOUT84	TIM0_7	Reserved	TOM0_7	TOM1_7	TOM0_7N	TOM1_7N
P14.6	TIN86	TOUT86	TIM0_1	Reserved	TOM0_1	TOM1_1	Reserved	Reserved
P15.0	Reserved	TOUT71	Reserved	Reserved	TOM1_3	TOM0_11	TOM0_7N	TOM1_7N
P15.1	Reserved	TOUT72	Reserved	Reserved	TOM1_4	TOM0_12	TOM0_4	TOM1_4
P15.2	Reserved	TOUT73	Reserved	Reserved	TOM1_5	TOM0_13	TOM0_4N	TOM1_4N
P15.3	Reserved	TOUT74	Reserved	Reserved	TOM1_6	TOM0_14	TOM0_5	TOM1_5
P15.5	Reserved	TOUT76	Reserved	Reserved	TOM0_0	TOM1_0	TOM0_5N	TOM1_5N
P20.8	TIN64	TOUT64	TIM0_7	Reserved	TOM1_7	TOM0_7	TOM0_4	TOM1_4
P20.9	Reserved	TOUT65	Reserved	Reserved	TOM1_13	TOM0_13	TOM0_4N	TOM1_4N

Table 24-28 GTM to Port Mapping for QFP-80

Port	Input	Output	Input Timer Mapped		Output Timer Mapped			
			A	B	A	B	C	D
P20.11	Reserved	TOUT67	Reserved	Reserved	TOM1_15	TOM0_15	TOM0_5N	TOM1_5N
P20.12	Reserved	TOUT68	Reserved	Reserved	TOM1_0	TOM0_8	TOM0_6	TOM1_6
P20.13	Reserved	TOUT69	Reserved	Reserved	TOM1_1	TOM0_9	TOM0_6N	TOM1_6N
P20.14	Reserved	TOUT70	Reserved	Reserved	TOM1_2	TOM0_10	TOM0_7	TOM1_7
P21.6	TIN57	TOUT57	TIM0_4	Reserved	TOM0_4	TOM1_4	Reserved	Reserved
P21.7	TIN58	TOUT58	TIM0_5	Reserved	TOM0_5	TOM1_5	Reserved	Reserved
P23.1	TIN42	TOUT42	TIM0_6	Reserved	TOM0_6	TOM0_15	Reserved	Reserved
P33.5	TIN27	TOUT27	TIM0_1	Reserved	TOM0_1	TOM1_1	TOM0_5	TOM1_5
P33.6	TIN28	TOUT28	TIM0_2	Reserved	TOM0_2	TOM1_2	TOM0_5N	TOM1_5N
P33.7	TIN29	TOUT29	TIM0_3	Reserved	TOM0_3	TOM1_3	TOM0_6	TOM1_6
P33.8	TIN30	TOUT30	TIM0_4	Reserved	TOM0_4	TOM1_4	TOM0_6N	TOM1_6N
P33.9	TIN31	TOUT31	TIM0_1	Reserved	TOM0_1	TOM1_1	TOM0_7	TOM1_7
P33.10	TIN32	TOUT32	TIM0_0	Reserved	TOM0_0	TOM1_0	TOM0_7N	TOM1_7N

Table 24-29 GTM to Port Mapping for QFP-100

Port	Input	Output	Input Timer Mapped		Output Timer Mapped			
			A	B	A	B	C	D
P00.0	TIN9	TOUT9	Reserved	TIM0_0	TOM0_8	TOM1_0	TOM0_4	TOM1_4
P02.0	TIN0	TOUT0	TIM0_0	Reserved	TOM0_8	TOM1_8	TOM0_4	TOM1_4
P02.1	TIN1	TOUT1	TIM0_1	Reserved	TOM0_9	TOM1_9	TOM0_4N	TOM1_4N
P02.2	TIN2	TOUT2	TIM0_2	Reserved	TOM0_10	TOM1_10	TOM0_5	TOM1_5
P02.3	TIN3	TOUT3	TIM0_3	Reserved	TOM0_11	TOM1_11	TOM0_5N	TOM1_5N
P02.4	TIN4	TOUT4	TIM0_4	Reserved	TOM0_12	TOM1_12	TOM0_6	TOM1_6
P02.5	TIN5	TOUT5	TIM0_5	Reserved	TOM0_13	TOM1_13	TOM0_6N	TOM1_6N
P02.6	TIN6	TOUT6	TIM0_6	Reserved	TOM0_14	TOM1_14	TOM0_7	TOM1_7
P02.7	TIN7	TOUT7	TIM0_7	Reserved	TOM0_15	TOM1_15	TOM0_7N	TOM1_7N
P02.8	TIN8	TOUT8	Reserved	TIM0_0	TOM0_8	TOM1_0	TOM0_4N	TOM1_4N
P10.5	TIN107	TOUT107	TIM0_2	Reserved	TOM0_2	TOM1_10	Reserved	Reserved
P10.6	TIN108	TOUT108	TIM0_3	Reserved	TOM0_3	TOM1_11	Reserved	Reserved
P11.2	Reserved	TOUT95	Reserved	Reserved	TOM0_8	TOM1_1	TOM0_4N	TOM1_4N
P11.3	Reserved	TOUT96	Reserved	Reserved	TOM0_10	TOM1_2	TOM0_5	TOM1_5
P11.6	Reserved	TOUT97	Reserved	Reserved	TOM0_11	TOM1_3	TOM0_5N	TOM1_5N

Table 24-29 GTM to Port Mapping for QFP-100

Port	Input	Output	Input Timer Mapped		Output Timer Mapped			
			A	B	A	B	C	D
P11.8	Reserved	TOUT110	Reserved	Reserved	Reserved	Reserved	TOM0_4	TOM1_4
P11.9	Reserved	TOUT98	Reserved	Reserved	TOM0_12	TOM1_4	TOM0_6	TOM1_6
P11.10	Reserved	TOUT99	Reserved	Reserved	TOM0_13	TOM1_5	TOM0_6N	TOM1_6N
P11.11	Reserved	TOUT100	Reserved	Reserved	TOM0_14	TOM1_6	TOM0_7N	TOM1_7N
P11.12	Reserved	TOUT101	Reserved	Reserved	TOM0_15	TOM1_7	TOM0_7	TOM1_7
P13.0	Reserved	TOUT91	Reserved	Reserved	TOM0_5	TOM1_5	TOM0_6N	TOM1_6N
P13.1	Reserved	TOUT92	Reserved	Reserved	TOM0_6	TOM1_6	TOM0_7	TOM1_7
P13.2	Reserved	TOUT93	Reserved	Reserved	TOM0_7	TOM1_7	TOM0_7N	TOM1_7N
P13.3	Reserved	TOUT94	Reserved	Reserved	TOM0_8	TOM1_0	TOM0_4	TOM1_4
P14.0	TIN80	TOUT80	TIM0_3	Reserved	TOM0_3	TOM1_3	TOM0_6	TOM1_6
P14.1	TIN81	TOUT81	TIM0_4	Reserved	TOM0_4	TOM1_4	TOM0_7	TOM1_7
P14.3	TIN83	TOUT83	TIM0_6	Reserved	TOM0_6	TOM1_6	Reserved	Reserved
P14.4	TIN84	TOUT84	TIM0_7	Reserved	TOM0_7	TOM1_7	TOM0_7N	TOM1_7N
P14.6	TIN86	TOUT86	TIM0_1	Reserved	TOM0_1	TOM1_1	Reserved	Reserved
P15.0	Reserved	TOUT71	Reserved	Reserved	TOM1_3	TOM0_11	TOM0_7N	TOM1_7N
P15.1	Reserved	TOUT72	Reserved	Reserved	TOM1_4	TOM0_12	TOM0_4	TOM1_4

Table 24-29 GTM to Port Mapping for QFP-100

Port	Input	Output	Input Timer Mapped		Output Timer Mapped			
			A	B	A	B	C	D
P15.2	Reserved	TOUT73	Reserved	Reserved	TOM1_5	TOM0_13	TOM0_4N	TOM1_4N
P15.3	Reserved	TOUT74	Reserved	Reserved	TOM1_6	TOM0_14	TOM0_5	TOM1_5
P15.5	Reserved	TOUT76	Reserved	Reserved	TOM0_0	TOM1_0	TOM0_5N	TOM1_5N
P20.8	TIN64	TOUT64	TIM0_7	Reserved	TOM1_7	TOM0_7	TOM0_4	TOM1_4
P20.9	Reserved	TOUT65	Reserved	Reserved	TOM1_13	TOM0_13	TOM0_4N	TOM1_4N
P20.10	Reserved	TOUT66	Reserved	Reserved	TOM1_14	TOM0_14	TOM0_5	TOM1_5
P20.11	Reserved	TOUT67	Reserved	Reserved	TOM1_15	TOM0_15	TOM0_5N	TOM1_5N
P20.12	Reserved	TOUT68	Reserved	Reserved	TOM1_0	TOM0_8	TOM0_6	TOM1_6
P20.13	Reserved	TOUT69	Reserved	Reserved	TOM1_1	TOM0_9	TOM0_6N	TOM1_6N
P20.14	Reserved	TOUT70	Reserved	Reserved	TOM1_2	TOM0_10	TOM0_7	TOM1_7
P21.2	TIN53	TOUT53	TIM0_0	Reserved	TOM0_0	TOM1_0	TOM0_4	TOM1_4
P21.3	TIN54	TOUT54	TIM0_1	Reserved	TOM0_1	TOM1_1	TOM0_4N	TOM1_4N
P21.4	TIN55	TOUT55	TIM0_2	Reserved	TOM0_2	TOM1_2	TOM0_5	TOM1_5
P21.6	TIN57	TOUT57	TIM0_4	Reserved	TOM0_4	TOM1_4	Reserved	Reserved
P21.7	TIN58	TOUT58	TIM0_5	Reserved	TOM0_5	TOM1_5	Reserved	Reserved
P23.1	TIN42	TOUT42	TIM0_6	Reserved	TOM0_6	TOM0_15	Reserved	Reserved

Table 24-29 GTM to Port Mapping for QFP-100

Port	Input	Output	Input Timer Mapped		Output Timer Mapped			
			A	B	A	B	C	D
P33.5	TIN27	TOUT27	TIM0_1	Reserved	TOM0_1	TOM1_1	TOM0_5	TOM1_5
P33.6	TIN28	TOUT28	TIM0_2	Reserved	TOM0_2	TOM1_2	TOM0_5N	TOM1_5N
P33.7	TIN29	TOUT29	TIM0_3	Reserved	TOM0_3	TOM1_3	TOM0_6	TOM1_6
P33.8	TIN30	TOUT30	TIM0_4	Reserved	TOM0_4	TOM1_4	TOM0_6N	TOM1_6N
P33.9	TIN31	TOUT31	TIM0_1	Reserved	TOM0_1	TOM1_1	TOM0_7	TOM1_7
P33.10	TIN32	TOUT32	TIM0_0	Reserved	TOM0_0	TOM1_0	TOM0_7N	TOM1_7N

Table 24-30 GTM to Port Mapping for QFP-144 and BGA-292

Port	Input	Output	Input Timer Mapped		Output Timer Mapped			
			A	B	A	B	C	D
P00.0	TIN9	TOUT9	Reserved	TIM0_0	TOM0_8	TOM1_0	TOM0_4	TOM1_4
P00.1	TIN10	TOUT10	Reserved	TIM0_1	TOM0_9	TOM1_1	TOM0_4N	TOM1_4N
P00.2	TIN11	TOUT11	Reserved	TIM0_1	TOM0_9	TOM1_1	TOM0_5	TOM1_5
P00.3	Reserved	TOUT12	Reserved	Reserved	TOM0_10	TOM1_2	TOM0_5N	TOM1_5N
P00.4	Reserved	TOUT13	Reserved	Reserved	TOM0_11	TOM1_3	TOM0_6	TOM1_6
P00.5	Reserved	TOUT14	Reserved	Reserved	TOM0_12	TOM1_4	TOM0_6N	TOM1_6N
P00.6	Reserved	TOUT15	Reserved	Reserved	TOM0_13	TOM1_5	TOM0_7	TOM1_7
P00.7	Reserved	TOUT16	Reserved	Reserved	TOM0_14	TOM1_6	TOM0_7N	TOM1_7N
P00.8	Reserved	TOUT17	Reserved	Reserved	TOM0_15	TOM1_7	Reserved	Reserved
P00.9	TIN18	TOUT18	TIM0_0	Reserved	TOM0_0	TOM1_0	Reserved	Reserved
P00.12	TIN21	TOUT21	TIM0_3	Reserved	TOM0_3	TOM1_3	Reserved	Reserved
P02.0	TIN0	TOUT0	TIM0_0	Reserved	TOM0_8	TOM1_8	TOM0_4	TOM1_4
P02.1	TIN1	TOUT1	TIM0_1	Reserved	TOM0_9	TOM1_9	TOM0_4N	TOM1_4N
P02.2	TIN2	TOUT2	TIM0_2	Reserved	TOM0_10	TOM1_10	TOM0_5	TOM1_5
P02.3	TIN3	TOUT3	TIM0_3	Reserved	TOM0_11	TOM1_11	TOM0_5N	TOM1_5N

Table 24-30 GTM to Port Mapping for QFP-144 and BGA-292

Port	Input	Output	Input Timer Mapped		Output Timer Mapped			
			A	B	A	B	C	D
P02.4	TIN4	TOUT4	TIM0_4	Reserved	TOM0_12	TOM1_12	TOM0_6	TOM1_6
P02.5	TIN5	TOUT5	TIM0_5	Reserved	TOM0_13	TOM1_13	TOM0_6N	TOM1_6N
P02.6	TIN6	TOUT6	TIM0_6	Reserved	TOM0_14	TOM1_14	TOM0_7	TOM1_7
P02.7	TIN7	TOUT7	TIM0_7	Reserved	TOM0_15	TOM1_15	TOM0_7N	TOM1_7N
P02.8	TIN8	TOUT8	Reserved	TIM0_0	TOM0_8	TOM1_0	TOM0_4N	TOM1_4N
P10.1	TIN103	TOUT103	TIM0_1	Reserved	TOM0_1	TOM1_9	Reserved	Reserved
P10.2	TIN104	TOUT104	TIM0_2	Reserved	TOM0_2	TOM1_10	Reserved	Reserved
P10.3	TIN105	TOUT105	TIM0_3	Reserved	TOM0_3	TOM1_11	Reserved	Reserved
P10.5	TIN107	TOUT107	TIM0_2	Reserved	TOM0_2	TOM1_10	Reserved	Reserved
P10.6	TIN108	TOUT108	TIM0_3	Reserved	TOM0_3	TOM1_11	Reserved	Reserved
P11.2	Reserved	TOUT95	Reserved	Reserved	TOM0_8	TOM1_1	TOM0_4N	TOM1_4N
P11.3	Reserved	TOUT96	Reserved	Reserved	TOM0_10	TOM1_2	TOM0_5	TOM1_5
P11.6	Reserved	TOUT97	Reserved	Reserved	TOM0_11	TOM1_3	TOM0_5N	TOM1_5N
P11.8	Reserved	TOUT110	Reserved	Reserved	Reserved	Reserved	TOM0_4	TOM1_4
P11.9	Reserved	TOUT98	Reserved	Reserved	TOM0_12	TOM1_4	TOM0_6	TOM1_6
P11.10	Reserved	TOUT99	Reserved	Reserved	TOM0_13	TOM1_5	TOM0_6N	TOM1_6N

Table 24-30 GTM to Port Mapping for QFP-144 and BGA-292

Port	Input	Output	Input Timer Mapped		Output Timer Mapped			
			A	B	A	B	C	D
P11.11	Reserved	TOUT100	Reserved	Reserved	TOM0_14	TOM1_6	TOM0_7N	TOM1_7N
P11.12	Reserved	TOUT101	Reserved	Reserved	TOM0_15	TOM1_7	TOM0_7	TOM1_7
P13.0	Reserved	TOUT91	Reserved	Reserved	TOM0_5	TOM1_5	TOM0_6N	TOM1_6N
P13.1	Reserved	TOUT92	Reserved	Reserved	TOM0_6	TOM1_6	TOM0_7	TOM1_7
P13.2	Reserved	TOUT93	Reserved	Reserved	TOM0_7	TOM1_7	TOM0_7N	TOM1_7N
P13.3	Reserved	TOUT94	Reserved	Reserved	TOM0_8	TOM1_0	TOM0_4	TOM1_4
P14.0	TIN80	TOUT80	TIM0_3	Reserved	TOM0_3	TOM1_3	TOM0_6	TOM1_6
P14.1	TIN81	TOUT81	TIM0_4	Reserved	TOM0_4	TOM1_4	TOM0_7	TOM1_7
P14.2	TIN82	TOUT82	TIM0_5	Reserved	TOM0_5	TOM1_5	TOM0_6N	TOM1_6N
P14.3	TIN83	TOUT83	TIM0_6	Reserved	TOM0_6	TOM1_6	Reserved	Reserved
P14.4	TIN84	TOUT84	TIM0_7	Reserved	TOM0_7	TOM1_7	TOM0_7N	TOM1_7N
P14.5	TIN85	TOUT85	TIM0_0	Reserved	TOM0_0	TOM1_0	Reserved	Reserved
P14.6	TIN86	TOUT86	TIM0_1	Reserved	TOM0_1	TOM1_1	Reserved	Reserved
P14.7	TIN87	TOUT87	TIM0_0	Reserved	TOM0_0	Reserved	Reserved	Reserved
P14.8	Reserved	TOUT88	Reserved	Reserved	TOM0_2	Reserved	Reserved	Reserved
P15.0	Reserved	TOUT71	Reserved	Reserved	TOM1_3	TOM0_11	TOM0_7N	TOM1_7N

Table 24-30 GTM to Port Mapping for QFP-144 and BGA-292

Port	Input	Output	Input Timer Mapped		Output Timer Mapped			
			A	B	A	B	C	D
P15.1	Reserved	TOUT72	Reserved	Reserved	TOM1_4	TOM0_12	TOM0_4	TOM1_4
P15.2	Reserved	TOUT73	Reserved	Reserved	TOM1_5	TOM0_13	TOM0_4N	TOM1_4N
P15.3	Reserved	TOUT74	Reserved	Reserved	TOM1_6	TOM0_14	TOM0_5	TOM1_5
P15.4	Reserved	TOUT75	Reserved	Reserved	TOM1_7	TOM0_15	Reserved	Reserved
P15.5	Reserved	TOUT76	Reserved	Reserved	TOM0_0	TOM1_0	TOM0_5N	TOM1_5N
P15.6	TIN77	TOUT77	TIM0_0	Reserved	TOM0_0	TOM1_0	Reserved	Reserved
P15.7	TIN78	TOUT78	TIM0_1	Reserved	TOM0_1	TOM1_1	Reserved	Reserved
P15.8	TIN79	TOUT79	TIM0_2	Reserved	TOM0_2	TOM1_2	Reserved	Reserved
P20.0	TIN59	TOUT59	TIM0_6	Reserved	TOM0_6	TOM1_6	Reserved	Reserved
P20.3	TIN61	TOUT61	Reserved	TIM0_4	TOM1_12	TOM0_4	Reserved	Reserved
P20.6	TIN62	TOUT62	Reserved	TIM0_6	TOM1_10	TOM0_10	Reserved	Reserved
P20.7	TIN63	TOUT63	Reserved	TIM0_7	TOM1_11	TOM0_11	Reserved	Reserved
P20.8	TIN64	TOUT64	TIM0_7	Reserved	TOM1_7	TOM0_7	TOM0_4	TOM1_4
P20.9	Reserved	TOUT65	Reserved	Reserved	TOM1_13	TOM0_13	TOM0_4N	TOM1_4N
P20.10	Reserved	TOUT66	Reserved	Reserved	TOM1_14	TOM0_14	TOM0_5	TOM1_5
P20.11	Reserved	TOUT67	Reserved	Reserved	TOM1_15	TOM0_15	TOM0_5N	TOM1_5N

Table 24-30 GTM to Port Mapping for QFP-144 and BGA-292

Port	Input	Output	Input Timer Mapped		Output Timer Mapped			
			A	B	A	B	C	D
P20.12	Reserved	TOUT68	Reserved	Reserved	TOM1_0	TOM0_8	TOM0_6	TOM1_6
P20.13	Reserved	TOUT69	Reserved	Reserved	TOM1_1	TOM0_9	TOM0_6N	TOM1_6N
P20.14	Reserved	TOUT70	Reserved	Reserved	TOM1_2	TOM0_10	TOM0_7	TOM1_7
P21.2	TIN53	TOUT53	TIM0_0	Reserved	TOM0_0	TOM1_0	TOM0_4	TOM1_4
P21.3	TIN54	TOUT54	TIM0_1	Reserved	TOM0_1	TOM1_1	TOM0_4N	TOM1_4N
P21.4	TIN55	TOUT55	TIM0_2	Reserved	TOM0_2	TOM1_2	TOM0_5	TOM1_5
P21.5	TIN56	TOUT56	TIM0_3	Reserved	TOM0_3	TOM1_3	TOM0_5N	TOM1_5N
P21.6	TIN57	TOUT57	TIM0_4	Reserved	TOM0_4	TOM1_4	Reserved	Reserved
P21.7	TIN58	TOUT58	TIM0_5	Reserved	TOM0_5	TOM1_5	Reserved	Reserved
P22.0	TIN47	TOUT47	TIM0_1	Reserved	TOM0_9	TOM1_1	Reserved	Reserved
P22.1	TIN48	TOUT48	TIM0_0	Reserved	TOM0_8	TOM1_0	TOM0_6	TOM1_6
P22.2	TIN49	TOUT49	TIM0_3	Reserved	TOM0_11	TOM1_3	TOM0_6N	TOM1_6N
P22.3	TIN50	TOUT50	TIM0_4	Reserved	TOM0_12	TOM1_4	TOM0_7	TOM1_7
P22.4	Reserved	TOUT111	Reserved	Reserved	Reserved	Reserved	TOM0_7N	TOM1_7N
P23.1	TIN42	TOUT42	TIM0_6	Reserved	TOM0_6	TOM0_15	Reserved	Reserved
P33.0	TIN22	TOUT22	TIM0_4	Reserved	TOM0_4	TOM1_4	Reserved	Reserved

Table 24-30 GTM to Port Mapping for QFP-144 and BGA-292

Port	Input	Output	Input Timer Mapped		Output Timer Mapped			
			A	B	A	B	C	D
P33.1	TIN23	TOUT23	TIM0_5	Reserved	TOM0_5	TOM1_5	Reserved	Reserved
P33.2	TIN24	TOUT24	TIM0_6	Reserved	TOM0_6	TOM1_6	Reserved	Reserved
P33.3	TIN25	TOUT25	TIM0_7	Reserved	TOM0_7	TOM1_7	Reserved	Reserved
P33.4	TIN26	TOUT26	TIM0_0	Reserved	TOM0_0	TOM1_0	Reserved	Reserved
P33.5	TIN27	TOUT27	TIM0_1	Reserved	TOM0_1	TOM1_1	TOM0_5	TOM1_5
P33.6	TIN28	TOUT28	TIM0_2	Reserved	TOM0_2	TOM1_2	TOM0_5N	TOM1_5N
P33.7	TIN29	TOUT29	TIM0_3	Reserved	TOM0_3	TOM1_3	TOM0_6	TOM1_6
P33.8	TIN30	TOUT30	TIM0_4	Reserved	TOM0_4	TOM1_4	TOM0_6N	TOM1_6N
P33.9	TIN31	TOUT31	TIM0_1	Reserved	TOM0_1	TOM1_1	TOM0_7	TOM1_7
P33.10	TIN32	TOUT32	TIM0_0	Reserved	TOM0_0	TOM1_0	TOM0_7N	TOM1_7N
P33.11	TIN33	TOUT33	TIM0_2	Reserved	TOM0_2	TOM1_2	Reserved	Reserved
P33.12	TIN34	TOUT34	Reserved	TIM0_0	TOM1_12	TOM0_12	Reserved	Reserved
P34.0	Reserved	TOUT112	Reserved	Reserved	TOM1_12	Reserved	Reserved	Reserved
P34.1	Reserved	TOUT113	Reserved	Reserved	TOM1_13	Reserved	Reserved	Reserved
P34.2	Reserved	TOUT114	Reserved	Reserved	TOM1_14	Reserved	Reserved	Reserved
P34.3	Reserved	TOUT115	Reserved	Reserved	TOM1_15	Reserved	Reserved	Reserved

The GTM outputs *cmu_eclk[2:0]* are connected directly to dedicated ports.

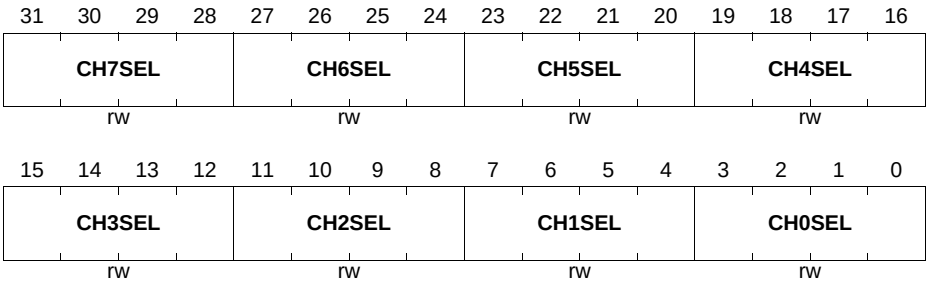
Table 24-31 GTM clock to Port Mapping

GTM Clock Output	Alternate Output of Pin
cmu_eclk0	P23.1
cmu_eclk1	P33.10
cmu_eclk2	P11.12

24.9.2.1 Port to GTM Control Registers

The inputs to the GTM TIM modules (*gtm_tim0_in[7:0]*) are not connected directly to the port input path. Each timed GPIO is connectable to two TIMs via an input multiplexer. In addition the inputs from the on chip peripherals are connected here too.

TIM0INSEL
TIM0 Input Select Register **($9FD10_H$)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CHxSEL (x = 0-7)	[x*4+3: x*4]	rw	**TIM Channel x Input Selection** This bit defines which input is connected for TIM0 channel x of the GTM. The input is either derived from the a port pad or from a on-chip module.

Note: Please note that for inputs from pins and for the ERAY input the TIM channel frequency needs to be equal or faster as the input frequency.

Table 24-32 TIM 0 Mapping for QFP-80

CH0SEL	Pad / Input	Name
0000_B	'0'	-
0001_B	Reserved	-
0010_B	P02.0	TIN0
0011_B	Reserved	-
0100_B	P02.8	TIN8
0101_B	P13.3	TIN94
0110_B	Reserved	-
0111_B	P21.2	TIN53
1000_B	P20.12	TIN68
1001_B	P33.10	TIN32
1010_B	Reserved	-
1011_B	Reserved	-
1100_B	P33.12	TIN34
1101_B	P00.0	TIN9
1110_B	Reserved	-
1111_B	*vadc_C0SR0*	ADC
CH1SEL	**Pad / Input**	**Name**
0000_B	'0'	-
0001_B	Reserved	-
0010_B	P02.1	TIN1
0011_B	P00.1	TIN10
0100_B	P00.2	TIN11
0101_B	P11.2	TIN95
0110_B	P21.3	TIN54
0111_B	Reserved	-
1000_B	Reserved	-
1001_B	P33.9	TIN31
1010_B	Reserved	-
1011_B	Reserved	-
1100_B	Reserved	-

Table 24-32 TIM 0 Mapping for QFP-80 (cont'd)

1101_B	*can_int[12]*	CAN
1110_B	Reserved	-
1111_B	*vadc_C1SR0*	ADC
CH2SEL	**Pad / Input**	**Name**
0000_B	'0'	-
0001_B	P00.3	TIN12
0010_B	P02.2	TIN2
0011_B	P11.3	TIN96
0100_B	Reserved	-
0101_B	P20.14	TIN70
0110_B	P21.4	TIN55
0111_B	Reserved	-
1000_B	P33.11	TIN33
1001_B	Reserved	-
1010_B	Reserved	-
1011_B	Reserved	-
1100_B	*eru_pdout2*	SCU
1101_B	*can_int[13]*	CAN
1110_B	Reserved	-
1111_B	*vadc_C0SR1*	ADC
CH3SEL	**Pad / Input**	**Name**
0000_B	'0'	-
0001_B	P00.4	TIN13
0010_B	P02.3	TIN3
0011_B	P11.6	TIN97
0100_B	Reserved	-
0101_B	P14.0	TIN80
0110_B	P15.0	TIN71
0111_B	Reserved	-
1000_B	Reserved	-
1001_B	Reserved	-

Table 24-32 TIM 0 Mapping for QFP-80 (cont'd)

1010_B	Reserved	-
1011_B	Reserved	-
1100_B	*eru_pdout3*	SCU
1101_B	*can_int[14]*	CAN
1110_B	Reserved	-
1111_B	*vadc_C1SR1*	ADC
CH4SEL	**Pad / Input**	**Name**
0000_B	'0'	-
0001_B	P02.4	TIN4
0010_B	P00.5	TIN14
0011_B	P14.1	TIN81
0100_B	P11.9	TIN98
0101_B	P15.1	TIN72
0110_B	Reserved	-
0111_B	P33.8	TIN30
1000_B	P21.6	TIN57
1001_B	Reserved	-
1010_B	Reserved	-
1011_B	Reserved	-
1100_B	*eru_pdout4*	SCU
1101_B	*can_int[15]*	CAN
1110_B	Reserved	-
1111_B	*vadc_C0SR2*	ADC
CH5SEL	**Pad / Input**	**Name**
0000_B	'0'	-
0001_B	P02.5	TIN5
0010_B	Reserved	-
0011_B	P11.10	TIN99
0100_B	P13.0	TIN91
0101_B	Reserved	-
0110_B	P15.2	TIN73

Table 24-32 TIM 0 Mapping for QFP-80 (cont'd)

0111_B	P21.7	TIN58
1000_B	Reserved	-
1001_B	Reserved	-
1010_B	Reserved	-
1011_B	Reserved	-
1100_B	*eru_pdout5*	SCU
1101_B	Reserved	-
1110_B	Reserved	-
1111_B	*vadc_C1SR2*	ADC
CH6SEL	**Pad / Input**	**Name**
0000_B	'0'	-
0001_B	P02.6	TIN6
0010_B	Reserved	-
0011_B	P14.3	TIN83
0100_B	P11.11	TIN100
0101_B	P13.1	TIN92
0110_B	P15.3	TIN74
0111_B	Reserved	-
1000_B	Reserved	-
1001_B	Reserved	-
1010_B	Reserved	-
1011_B	Reserved	-
1100_B	*eru_pdout6*	SCU
1101_B	Reserved	-
1110_B	Reserved	-
1111_B	*vadc_C0SR3*	ADC
CH7SEL	**Pad / Input**	**Name**
0000_B	'0'	-
0001_B	P02.7	TIN7
0010_B	Reserved	-
0011_B	Reserved	-

Table 24-32 TIM 0 Mapping for QFP-80 (cont'd)

0100_B	P08.8	TIN17
0101_B	P11.12	TIN101
0110_B	P20.11	TIN67
0111_B	Reserved	-
1000_B	Reserved	-
1001_B	Reserved	-
1010_B	Reserved	-
1011_B	Reserved	-
1100_B	*eru_pdout7*	SCU
1101_B	Reserved	-
1110_B	Reserved	-
1111_B	*vadc_C1SR3*	ADC

Table 24-33 TIM 0 Mapping for QFP-100

CH0SEL	Pad / Input	Name
0000_B	'0'	-
0001_B	P00.9	TIN18
0010_B	P02.0	TIN0
0011_B	Reserved	-
0100_B	Reserved	-
0101_B	Reserved	-
0110_B	Reserved	-
0111_B	P21.2	TIN53
1000_B	Reserved	-
1001_B	P33.10	TIN32
1010_B	P33.4	TIN26
1011_B	P02.8	TIN8
1100_B	P33.12	TIN34
1101_B	P00.0	TIN9
1110_B	Reserved	-
1111_B	*vadc_C0SR0*	ADC
CH1SEL	**Pad / Input**	**Name**
0000_B	'0'	-
0001_B	Reserved	-
0010_B	P02.1	TIN1
0011_B	Reserved	-
0100_B	P14.6	TIN86
0101_B	Reserved	-
0110_B	P21.3	TIN54
0111_B	Reserved	-
1000_B	P33.5	TIN27
1001_B	P33.9	TIN31
1010_B	Reserved	-
1011_B	Reserved	-
1100_B	Reserved	-

Table 24-33 TIM 0 Mapping for QFP-100 (cont'd)

1101_B	*can_int[12]*	CAN
1110_B	Reserved	-
1111_B	*vadc_C1SR0*	ADC
CH2SEL	**Pad / Input**	**Name**
0000_B	'0'	-
0001_B	Reserved	-
0010_B	P02.2	TIN2
0011_B	Reserved	-
0100_B	P10.5	TIN107
0101_B	Reserved	-
0110_B	P21.4	TIN55
0111_B	Reserved	-
1000_B	P33.11	TIN33
1001_B	P33.6	TIN28
1010_B	Reserved	-
1011_B	Reserved	-
1100_B	*eru_pdout2*	SCU
1101_B	*can_int[13]*	CAN
1110_B	Reserved	-
1111_B	*vadc_C0SR1*	ADC
CH3SEL	**Pad / Input**	**Name**
0000_B	'0'	-
0001_B	Reserved	-
0010_B	P02.3	TIN3
0011_B	Reserved	-
0100_B	P10.6	TIN108
0101_B	P14.0	TIN80
0110_B	Reserved	-
0111_B	Reserved	-
1000_B	Reserved	-
1001_B	P33.7	TIN29

Table 24-33 TIM 0 Mapping for QFP-100 (cont'd)

1010_B	Reserved	-
1011_B	Reserved	-
1100_B	*eru_pdout3*	SCU
1101_B	*can_int[14]*	CAN
1110_B	Reserved	-
1111_B	*vadc_C1SR1*	ADC
CH4SEL	**Pad / Input**	**Name**
0000_B	'0'	-
0001_B	P02.4	TIN4
0010_B	Reserved	-
0011_B	P14.1	TIN81
0100_B	Reserved	-
0101_B	Reserved	-
0110_B	Reserved	-
0111_B	P33.8	TIN30
1000_B	P21.6	TIN57
1001_B	P20.3	TIN61
1010_B	P33.0	TIN22
1011_B	Reserved	-
1100_B	*eru_pdout4*	SCU
1101_B	*can_int[15]*	CAN
1110_B	Reserved	-
1111_B	*vadc_C0SR2*	ADC
CH5SEL	**Pad / Input**	**Name**
0000_B	'0'	-
0001_B	P02.5	TIN5
0010_B	Reserved	-
0011_B	Reserved	-
0100_B	Reserved	-
0101_B	Reserved	-
0110_B	Reserved	-

Table 24-33 TIM 0 Mapping for QFP-100 (cont'd)

0111_B	P21.7	TIN58
1000_B	Reserved	-
1001_B	Reserved	-
1010_B	Reserved	-
1011_B	Reserved	-
1100_B	*eru_pdout5*	SCU
1101_B	Reserved	-
1110_B	Reserved	-
1111_B	*vadc_C1SR2*	ADC
CH6SEL	**Pad / Input**	**Name**
0000_B	'0'	-
0001_B	P02.6	TIN6
0010_B	Reserved	-
0011_B	P14.3	TIN83
0100_B	P23.1	TIN42
0101_B	Reserved	-
0110_B	Reserved	-
0111_B	Reserved	-
1000_B	Reserved	-
1001_B	Reserved	-
1010_B	Reserved	-
1011_B	Reserved	-
1100_B	*eru_pdout6*	SCU
1101_B	Reserved	-
1110_B	Reserved	-
1111_B	*vadc_C0SR3*	ADC
CH7SEL	**Pad / Input**	**Name**
0000_B	'0'	-
0001_B	P02.7	TIN7
0010_B	P14.4	TIN84
0011_B	P20.8	TIN64

Table 24-33 TIM 0 Mapping for QFP-100 (cont'd)

0100_B	Reserved	-
0101_B	Reserved	-
0110_B	Reserved	-
0111_B	Reserved	-
1000_B	Reserved	-
1001_B	Reserved	-
1010_B	Reserved	-
1011_B	Reserved	-
1100_B	*eru_pdout7*	SCU
1101_B	*eray_mt*	ERAY-
1110_B	Reserved	-
1111_B	*vadc_C1SR3*	ADC

Table 24-34 TIM 0 Mapping for QFP-144 / BGA-292

CH0SEL	Pad / Input	Name
0000_B	'0'	-
0001_B	P00.9	TIN18
0010_B	P02.0	TIN0
0011_B	Reserved	-
0100_B	P14.5	TIN85
0101_B	P14.7	TIN87
0110_B	P15.6	TIN77
0111_B	P21.2	TIN53
1000_B	P22.1	TIN48
1001_B	P33.10	TIN32
1010_B	P33.4	TIN26
1011_B	P02.8	TIN8
1100_B	P33.12	TIN34
1101_B	P00.0	TIN9
1110_B	Reserved	-
1111_B	*vadc_C0SR0*	ADC
CH1SEL	**Pad / Input**	**Name**
0000_B	'0'	-
0001_B	Reserved	-
0010_B	P02.1	TIN1
0011_B	P10.1	TIN103
0100_B	P14.6	TIN86
0101_B	P15.7	TIN78
0110_B	P21.3	TIN54
0111_B	P22.0	TIN47
1000_B	P33.5	TIN27
1001_B	P33.9	TIN31
1010_B	P00.1	TIN10
1011_B	P00.2	TIN11
1100_B	Reserved	-

Table 24-34 TIM 0 Mapping for QFP-144 (cont'd)**/ BGA-292**

1101_B	*can_int[12]*	CAN
1110_B	Reserved	-
1111_B	*vadc_C1SR0*	ADC
CH2SEL	**Pad / Input**	**Name**
0000_B	'0'	-
0001_B	Reserved	-
0010_B	P02.2	TIN2
0011_B	P10.2	TIN104
0100_B	P10.5	TIN107
0101_B	P15.8	TIN79
0110_B	P21.4	TIN55
0111_B	Reserved	-
1000_B	P33.11	TIN33
1001_B	P33.6	TIN28
1010_B	Reserved	-
1011_B	Reserved	-
1100_B	*eru_pdout2*	SCU
1101_B	*can_int[13]*	CAN
1110_B	Reserved	-
1111_B	*vadc_C0SR1*	ADC
CH3SEL	**Pad / Input**	**Name**
0000_B	'0'	-
0001_B	P00.12	TIN21
0010_B	P02.3	TIN3
0011_B	P10.3	TIN105
0100_B	P10.6	TIN108
0101_B	P14.0	TIN80
0110_B	P21.5	TIN56
0111_B	P22.2	TIN49
1000_B	Reserved	-
1001_B	P33.7	TIN29

Table 24-34 TIM 0 Mapping for QFP-144 (cont'd)**/ BGA-292**

1010_B	Reserved	-
1011_B	Reserved	-
1100_B	*eru_pdout3*	SCU
1101_B	*can_int[14]*	CAN
1110_B	Reserved	-
1111_B	*vadc_C1SR1*	ADC
CH4SEL	**Pad / Input**	**Name**
0000_B	'0'	-
0001_B	P02.4	TIN4
0010_B	Reserved	-
0011_B	P14.1	TIN81
0100_B	P22.3	TIN50
0101_B	Reserved	-
0110_B	P33.0	TIN22
0111_B	P33.8	TIN30
1000_B	P21.6	TIN57
1001_B	P20.3	TIN61
1010_B	Reserved	-
1011_B	Reserved	-
1100_B	*eru_pdout4*	SCU
1101_B	*can_int[15]*	CAN
1110_B	Reserved	-
1111_B	*vadc_C0SR2*	ADC
CH5SEL	**Pad / Input**	**Name**
0000_B	'0'	-
0001_B	P02.5	TIN5
0010_B	Reserved	-
0011_B	P14.2	TIN82
0100_B	Reserved	-
0101_B	Reserved	-
0110_B	P33.1	TIN23

Table 24-34 TIM 0 Mapping for QFP-144 (cont'd)**/ BGA-292**

0111_B	P21.7	TIN58
1000_B	Reserved	-
1001_B	Reserved	-
1010_B	Reserved	-
1011_B	Reserved	-
1100_B	*eru_pdout5*	SCU
1101_B	Reserved	-
1110_B	Reserved	-
1111_B	*vadc_C1SR2*	ADC
CH6SEL	**Pad / Input**	**Name**
0000_B	'0'	-
0001_B	P02.6	TIN6
0010_B	Reserved	-
0011_B	P14.3	TIN83
0100_B	P23.1	TIN42
0101_B	Reserved	-
0110_B	P33.2	TIN24
0111_B	P20.0	TIN59
1000_B	Reserved	-
1001_B	P20.6	TIN62
1010_B	Reserved	-
1011_B	Reserved	-
1100_B	*eru_pdout6*	SCU
1101_B	Reserved	-
1110_B	Reserved	-
1111_B	*vadc_C0SR3*	ADC
CH7SEL	**Pad / Input**	**Name**
0000_B	'0'	-
0001_B	P02.7	TIN7
0010_B	P14.4	TIN84
0011_B	P20.8	TIN64

Table 24-34 TIM 0 Mapping for QFP-144 (cont'd)**/ BGA-292**

0100_B	Reserved	-
0101_B	Reserved	-
0110_B	P33.3	TIN25
0111_B	P20.7	TIN63
1000_B	Reserved	-
1001_B	Reserved	-
1010_B	Reserved	-
1011_B	Reserved	-
1100_B	*eru_pdout7*	SCU
1101_B	*eray_mt*	ERAY-
1110_B	Reserved	-
1111_B	*vadc_C1SR3*	ADC

24.9.2.2 GTM to Port Control Registers

The outputs of the GTM TOM or DTM modules are not connected directly to the port output path. Each DTM is connectable to several port alternate inputs via an output multiplexer.

The output signals to the mux are called *gtm_tout_x*, where x is running number.

TOUTSELn (n = 0-7)
Timer Output Select Register (9FD30$_H$+n*4$_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEL15		SEL14		SEL13		SEL12		SEL11		SEL10		SEL9		SEL8	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEL7		SEL6		SEL5		SEL4		SEL3		SEL2		SEL1		SEL0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
SELx (x = 0-15)	[x*2+1: x*2]	rw	**TOUT(n*16+x) Output Selection** This bit defines which timer out is connected as TOUT(n*16+x). The mapping for each pin is defined by **Table 24-28**, **Table 24-29**, **Table 24-30**. 00_B Timer A form **Table 24-28**, **Table 24-29**, **Table 24-30** is connected as TOUT(n*16+x) to the ports 01_B Timer B form **Table 24-28**, **Table 24-29**, **Table 24-30** is connected as TOUT(n*16+x) to the ports 10_B Timer C form **Table 24-28**, **Table 24-29**, **Table 24-30** is connected as TOUT(n*16+x) to the ports 11_B Timer D form **Table 24-28**, **Table 24-29**, **Table 24-30** is connected as TOUT(n*16+x) to the ports *Note: If TOUT(n*16+x) is not defined in* **Table 24-28**, ***Table 24-29***, ***Table 24-30****this bit field has to be treated as reserved.*

24.9.3 ADC Connections

This register defines the GTM to ADC connections of TC21x/TC22x/TC23x.
The ADC trigger inputs 16 to 1 multiplexer is implemented twice per ADC module / channel. The output signal to the ADC is called *adcx_trig0* and *adcx_trig1*, The inputs of the multiplexer are called *adc_x_muxin[7:0]*, x = number of ADCs.

Mapping of ADC Triggers

Table 24-35 ADC0 / 1 Timer Triggers

SEL0 / 1	Trigger 0	Trigger 1
0000_B	No trigger is generated	No trigger is generated
0001_B	TOM0 Channel 1 output	TOM1 Channel 1 output
0010_B	TOM0 Channel 2 output	TOM1 Channel 2 output
0011_B	TOM0 Channel 3 output	TOM1 Channel 3 output
0100_B	TOM0 Channel 4 output	TOM1 Channel 4 output
0101_B	TOM0 Channel 5 output	TOM1 Channel 5 output
0110_B	TOM0 Channel 6 output	TOM1 Channel 6 output
0111_B	TOM0 Channel 7 output	TOM1 Channel 7 output
1000_B	TOM0 Channel 8 output	TOM1 Channel 8 output
1001_B	TOM0 Channel 9 output	TOM1 Channel 9 output
1010_B	TOM0 Channel 10 output	TOM1 Channel 10 output
1011_B	TOM0 Channel 11 output	TOM1 Channel 11 output
1100_B	TOM0 Channel 12 output	TOM1 Channel 12 output
1101_B	TOM0 Channel 13 output	TOM1 Channel 13 output
1110_B	TOM0 Channel 14 output	TOM1 Channel 14 output
1111_B	TOM0 Channel 15 output	TOM1 Channel 15 output

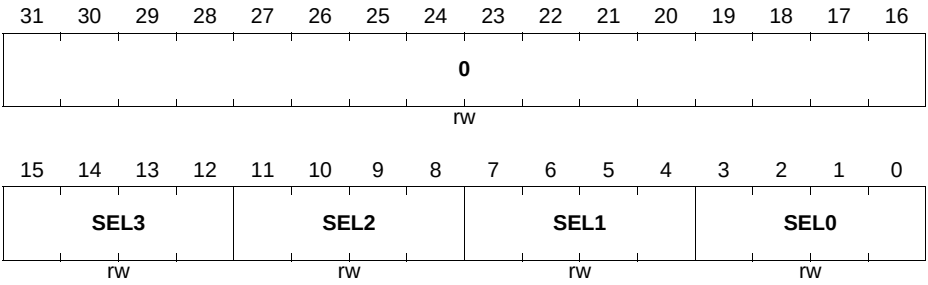
Table 24-36 ADC2 / 3 Timer Triggers

SEL3	Trigger 0	Trigger 1
0000_B	No trigger is generated	No trigger is generated
0001_B	TOM0 Channel 1 output	TOM1 Channel 1 output
0010_B	TOM0 Channel 2 output	TOM1 Channel 2 output
0011_B	TOM0 Channel 3 output	TOM1 Channel 3 output
0100_B	TOM0 Channel 4 output	TOM1 Channel 4 output

Table 24-36 ADC2 / 3 Timer Triggers (cont'd)

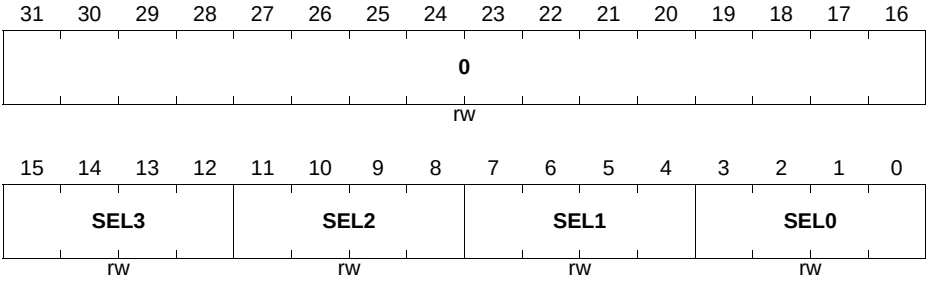
SEL3	Trigger 0	Trigger 1
0101_B	TOM0 Channel 5 output	TOM1 Channel 5 output
0110_B	TOM0 Channel 6 output	TOM1 Channel 6 output
0111_B	TOM0 Channel 7 output	TOM1 Channel 7 output
1000_B	TOM0 Channel 8 output	TOM1 Channel 8 output
1001_B	TOM0 Channel 9 output	TOM1 Channel 9 output
1010_B	TOM0 Channel 10 output	TOM1 Channel 10 output
1011_B	TOM0 Channel 11 output	TOM1 Channel 11 output
1100_B	TOM0 Channel 12 output	TOM1 Channel 12 output
1101_B	TOM0 Channel 13 output	TOM1 Channel 13 output
1110_B	TOM0 Channel 14 output	TOM1 Channel 14 output
1111_B	TOM0 Channel 15 output	TOM1 Channel 15 output

ADCTRIG0OUT0
ADC Trigger 0 Output Select 0 Register($9FDB0_H$) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
SELx (x = 0-3)	[x*4+3: x*4]	rw	**Output Selection for ADCx GTM connection** This bit field defines which TOM channel output is used as ADCx trigger 0. The decoding is defined by **Table 24-35** or **Table 24-36** depending on the ADC.
0	[31:16]	rw	**Reserved** Should be written with 0.

ADCTRIG1OUT0
ADC Trigger 1 Output Select 0 Register(9FDB8$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
SELx (x = 0-3)	[x*4+3: x*4]	rw	**Output Selection for ADCx GTM connection** This bit field defines which TOM channel output is used as ADCx trigger 1. The decoding is defined by **Table 24-35** depending on the ADC.
0	[31:16]	rw	**Reserved** Should be written with 0.

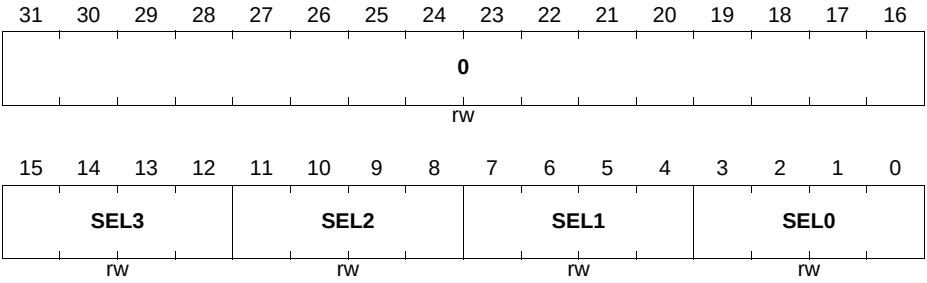
24.9.4 SENT Connections

This register defines the GTM to SENT connections of TC21x/TC22x/TC23x.
The trigger generation 16 to 1 multiplexer is implemented twice per ADC module / channel. The output signal to the ADC is called *adcx_trig0* and *adcx_trig1*. As the SENT channels overlay and replace ADC channels the ADC triggers will be also reused for the SENT channels. Therefore no additional outputs for separate operation are defined here.

adcx_trig0 is connected to sent_trig_x.

24.9.5 CAN Connection

CANOUTSEL
CAN Output Select Register (9FDA0$_H$) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
SEL0	[3:0]	rw	**Output Selection for CAN GTM connection** This bit field defines which TOM channel output is used as CAN node trigger 0. The decoding is defined by **Table 24-37**
SEL1	[7:4]	rw	**Output Selection for CAN GTM connection** This bit field defines which TOM channel output is used as CAN node trigger 1. The decoding is defined by **Table 24-37**
SEL2	[11:8]	rw	**Output Selection for CAN GTM connection** This bit field defines which TOM channel output is used as CAN node trigger 2. The decoding is defined by **Table 24-38**
SEL3	[15:12]	rw	**Output Selection for CAN GTM connection** This bit field defines which TOM channel output is used as CAN node trigger 3. The decoding is defined by **Table 24-38**
0	[31:16]	rw	**Reserved** Should be written with 0.

Mapping of CAN Triggers

Table 24-37 CAN Timer Triggers

SEL0 / 1	Trigger 0	Trigger 1
0000_B	TOM0 Channel 0 output	TOM1 Channel 0 output
0001_B	TOM0 Channel 1 output	TOM1 Channel 1 output
0010_B	TOM0 Channel 2 output	TOM1 Channel 2 output
0011_B	TOM0 Channel 3 output	TOM1 Channel 3 output
0100_B	TOM0 Channel 4 output	TOM1 Channel 4 output
0101_B	TOM0 Channel 5 output	TOM1 Channel 5 output
0110_B	TOM0 Channel 6 output	TOM1 Channel 6 output
0111_B	TOM0 Channel 7 output	TOM1 Channel 7 output
1000_B	TOM0 Channel 8 output	TOM1 Channel 8 output
1001_B	TOM0 Channel 9 output	TOM1 Channel 9 output
1010_B	TOM0 Channel 10 output	TOM1 Channel 10 output
1011_B	TOM0 Channel 11 output	TOM1 Channel 11 output
1100_B	TOM0 Channel 12 output	TOM1 Channel 12 output
1101_B	TOM0 Channel 13 output	TOM1 Channel 13 output
1110_B	TOM0 Channel 14 output	TOM1 Channel 14 output
1111_B	TOM0 Channel 15 output	TOM1 Channel 15 output

Table 24-38 CAN Timer Triggers

SEL2 / 3	Trigger 2	Trigger 3
0000_B	TOM0 Channel 0 output	TOM1 Channel 0 output
0001_B	TOM0 Channel 1 output	TOM1 Channel 1 output
0010_B	TOM0 Channel 2 output	TOM1 Channel 2 output
0011_B	TOM0 Channel 3 output	TOM1 Channel 3 output
0100_B	TOM0 Channel 4 output	TOM1 Channel 4 output
0101_B	TOM0 Channel 5 output	TOM1 Channel 5 output
0110_B	TOM0 Channel 6 output	TOM1 Channel 6 output
0111_B	TOM0 Channel 7 output	TOM1 Channel 7 output
1000_B	TOM0 Channel 8 output	TOM1 Channel 8 output
1001_B	TOM0 Channel 9 output	TOM1 Channel 9 output

Table 24-38 CAN Timer Triggers (cont'd)

SEL2 / 3	Trigger 2	Trigger 3
1010_B	TOM0 Channel 10 output	TOM1 Channel 10 output
1011_B	TOM0 Channel 11 output	TOM1 Channel 11 output
1100_B	TOM0 Channel 12 output	TOM1 Channel 12 output
1101_B	TOM0 Channel 13 output	TOM1 Channel 13 output
1110_B	TOM0 Channel 14 output	TOM1 Channel 14 output
1111_B	TOM0 Channel 15 output	TOM1 Channel 15 output

24.9.6 CCU6x Connections

This section summarize the connections to the CCU6x.

Table 24-39 GTM to CCU60 Connections

GTM Output	CCU60 Input
TOM0_8	T12HRD
TOM1_8	T13HRD

Table 24-40 GTM to CCU61 Connections

GTM Output	CCU61 Input
TOM0_9	T13HRD

24.9.7 SCU Connections

This section summarize the connections to the SCU.

Table 24-41 GTM to SCU Connections

GTM Output	SCU Input
TOM0_12	Input41
TOM1_12	Input51

24.9.8 GTM Debug Interface

OTGB0/1 are 16-bit trigger busses, OTGB2 is 32 bit wide. Further information is available in the OTGM (OCDS Trigger Mux) section of the device specification.

Note: If triggering or tracing is to be used the GTM clock must not be faster than the SPB clock.

24.9.8.1 OCDS Trigger Bus (OTGB) Interface

GTM OTGB Features

- IO signals
  - TIM input signals (after filter)
  - TOM output signals
  - Groups of 8 signals (e.g. all inputs of a specific TIM, etc.)
  - Two arbitrary groups on one 16 bit OTGB0/1
  - OTGB0 and/or OTGB1 can be used for 2-4 groups in parallel (e.g. both with one TIM 8 and one TOM 16)
- TBUs
  - Trace one selected TBU
  - Trigger signals for individual TBU comparators

The GTM module has one 16 bit and one 32 bit Trigger Sets (**Table 24-42**) which are selected with the **OTSS** register.

Table 24-42 GTM Trigger Sets

Trigger Set	Details
TS16_IOS Trigger Set IO and Other Signals	**Table 24-44**
TS32_TTB0/1/2 TBU Time Stamps (3 Sets)	**Table 24-45**

Table 24-43 shows all sensible Trigger Set mapping options. Simple permutations of OTGB0/1 are omitted. If OTGB0 and/or OTGB1 is not used for GTM, Trigger Sets of other sources can be added in the OTGM module. The Trigger Sets which will initiate the writing of a 32 or 64 bit data word to the MCDS trace are marked with bold letters. This is by design always the case for OTGB2 Trigger Sets. Independent OTGB0/1 Trigger Sets need an independent own sensitivity to their signal changes, configured in the OTGM module.

Table 24-43 Trigger Set Mapping Options

Width	OTGB0	OTGB1	OTGB2
32 Bit	**TS16_IOS**		
	TS16_IOS	**TS16_IOS**	
64 Bit			**TTB0/1/2**
	TS16_IOS		
	TS16_IOS	**TS16_IOS**	
		TS16_IOS	
		TS16_IOS	

The GTM trigger signals relate to the GTM internal clock, which can be slower or equal to the OTGB/OTGM clock.

IO and Other Signals Trigger Sets

IO Trigger Sets consist of the most important TIM and TOM signals in groups of 8. In addition there is a group of miscellaneous signals. The multiplexer allows to map arbitrary and different signal groups to the high and the low byte of a 16 bit Trigger Set. In addition it is possible to use one or two 16 bit Trigger Sets with this flexibility. All this is controlled with **OTSC0**.

Table 24-44 TS16_IOS Trigger Set IO and Other Signals

Bits	Name	Description	
[7:0]	SG0	A group of 8 signals from a selected TIM and TOM	
		TIM	All 8 input signals after filter F_OUT
		TOML	Lower 8 output signals TOM_OUT
		TOMH	Higher 8 output signals TOM_OUT
		MISC	Bit 0: reserved Bit 1: reserved Bit 4: TBU0 trigger (OTBU0T) Bit 5: TBU1 trigger (OTBU1T) Bit 6: TBU2 trigger (OTBU2T) Others: reserved
[15:8]	SG1	Independent selection with same options as for Bits [7:0]	

TBU Trigger Sets

Table 24-45 defines the Trigger Sets for the observation of the different time bases. The Trigger Set selection (one at a time) is done with OTSS.

Table 24-45 TS32_TTB0/1/2 TBU Time Stamps

Bits	Name	Description
[26:0]	TS	Current TBU_TS0/1/2 time stamp. Effective width depends on selected time base.
[31:27]		Reserved

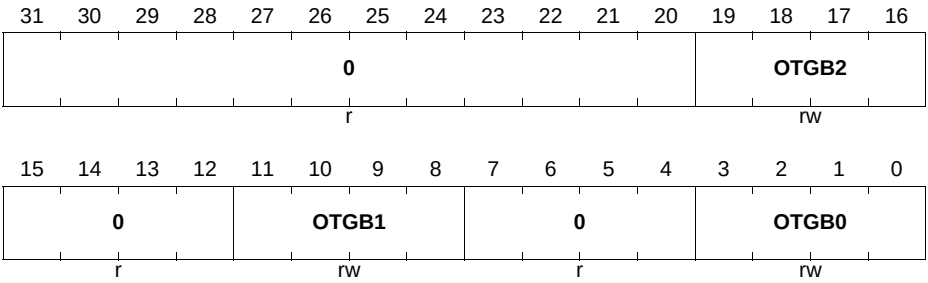
24.9.8.2 GTM Debug Registers

All debug control registers are cleared by Debug Reset and by each System Reset when OCDS is disabled. It is not changed by System Reset when OCDS is enabled.

OCDS Trigger Bus (OTGB)

Access are only supported for byte, halfword and word data and requires Supervisor Mode.

OTSS
OCDS Trigger Set Select Register (9FDD0$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
OTGB0	[3:0]	rw	**Trigger Set for OTGB0** 0_D No Trigger Set selected 1_D Trigger Set TS16_IOS (**Table 24-44**) **others**, reserved
OTGB1	[11:8]	rw	**Trigger Set for OTGB1** 0_D No Trigger Set selected 1_D Trigger Set TS16_IOS (**Table 24-44**) **others**, reserved
OTGB2	[19:16]	rw	**Trigger Set for OTGB2** 0_H No Trigger Set selected 8_H Trigger Set TS32_TTB0 (**Table 24-45**) 9_H Trigger Set TS32_TTB1 (**Table 24-45**) A_H Trigger Set TS32_TTB2 (**Table 24-45**) **others**, reserved
0	[7:4], [15:12], [31:20]	r	**Reserved** Read as 0; must be written with 0.

OTSC0

OCDS Trigger Set Control 0 Register(9FDD4$_H$) **Reset Value: 0000 0000$_H$**

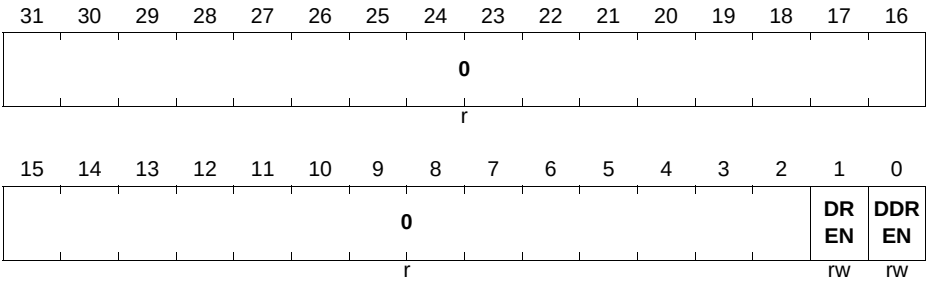
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
B1HMI				0	B1HMT			B1LMI				0	B1LMT		
rw				r	rw			rw				r	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B0HMI				0	B0HMT			B0LMI				0	B0LMT		
rw				r	rw			rw				r	rw		

Field	Bits	Type	Description
B0LMT	[2:0]	rw	**OTGB0 TS16_IOS Low Byte Module Type** 0_D No Module selected 1_D TIM (F_OUT[7:0]) 2_D TOML (TOM_OUT[7:0]) 3_D TOMH (TOM_OUT[15:8]) 6_D MISC signals (**Table 24-44**) **others**, reserved
B0LMI	[7:4]	rw	**OTGB0 TS16_IOS Low Byte Module Instance** Index of the module instance. Index starts with 0, the max. value depends on GTM configuration and module type. For MISC signals this index is ignored.
B0HMT	[10:8]	rw	**OTGB0 TS16_IOS High Byte Module Type** 0_D No Module selected 1_D TIM (F_OUT[7:0]) 2_D TOML (TOM_OUT[7:0]) 3_D TOMH (TOM_OUT[15:8]) 6_D MISC signals (**Table 24-44**) **others**, reserved
B0HMI	[15:12]	rw	**OTGB0 TS16_IOS High Byte Module Instance** Index of the module instance. Index starts with 0, the max. value depends on GTM configuration and module type. For MISC signals this index is ignored.

Field	Bits	Type	Description
B1LMT	[18:16]	rw	**OTGB1 TS16_IOS Low Byte Module Type** 0_D No Module selected 1_D TIM (F_OUT[7:0]) 2_D TOML (TOM_OUT[7:0]) 3_D TOMH (TOM_OUT[15:8]) 6_D MISC signals (**Table 24-44**) **others**, reserved
B1LMI	[23:20]	rw	**OTGB1 TS16_IOS Low Byte Module Instance** Index of the module instance. Index starts with 0, the max. value depends on GTM configuration and module type. For MISC signals this index is ignored.
B1HMT	[26:24]	rw	**OTGB1 TS16_IOS High Byte Module Type** 0_D No Module selected 1_D TIM (F_OUT[7:0]) 2_D TOML (TOM_OUT[7:0]) 3_D TOMH (TOM_OUT[15:8]) 6_D MISC signals (**Table 24-44**) **others**, reserved
B1HMI	[31:28]	rw	**OTGB1 TS16_IOS High Byte Module Instance** Index of the module instance. Index starts with 0, the max. value depends on GTM configuration and module type. For MISC signals this index is ignored.
0	3, 11,19,27	r	**Reserved** Read as 0; must be written with 0.

ODA
OCDS Debug Access Register (9FDDC$_H$) Reset Value: 0000 0000$_H$

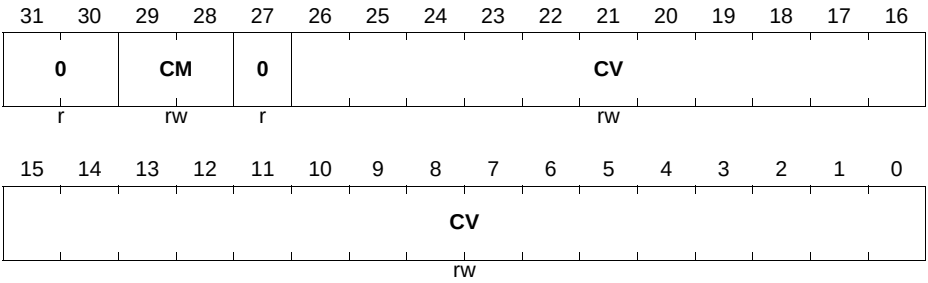


Field	Bits	Type	Description
DDREN	0	rw	**Destructive Debug Read Enable** For details see**Table 24-46**.
DREN	1	rw	**Destructive Read Enable** For details see**Table 24-46**.
0	[31:2]	r	**Reserved** Read as 0; should be written with 0.

Table 24-46 Destructive Read Control encoding

ODA.DREN	ODA.DDREN	Description
0	0	Destructive read access are enabled for all masters beside the OCDS master
0	1	Destructive read access are enabled for all masters
1	0	Destructive read access are disabled for all masters
1	1	Destructive read access are disabled for all masters

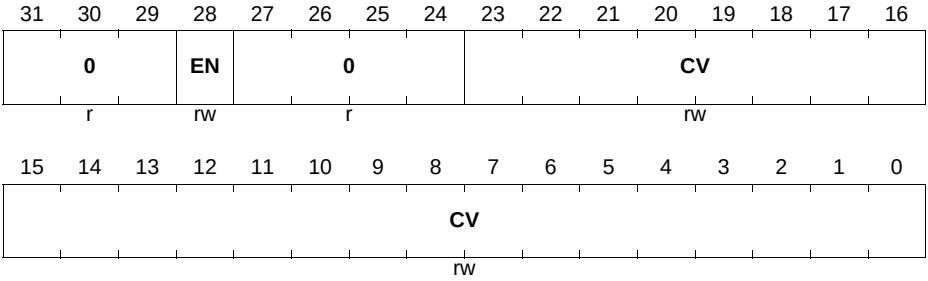
OTBU0T
OCDS TBU0 Trigger Register (9FDC4$_H$) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
CV	[26:0]	rw	**Compare Value** This value is compared to TBU_CH0_BASE register. As long as both match the associated TS16_IOS.MISC bit (**Table 24-44**) is active.

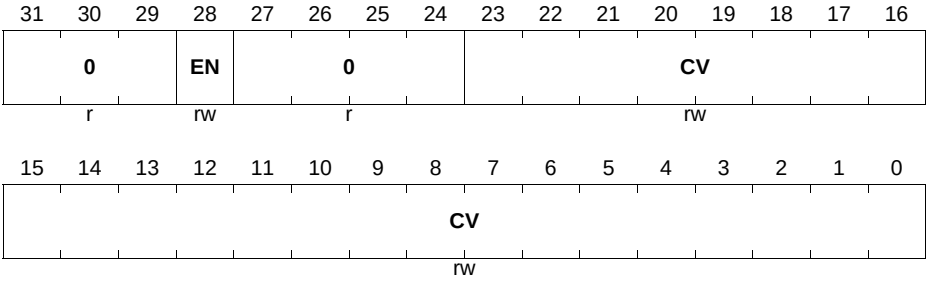
Field	Bits	Type	Description
CM	[29:28]	rw	**Compare Mode** 0_H Disabled 1_H Compare lower 24 bits 2_H Compare upper 24 bits 3_H Compare all 27 bits
0	27, [31:30]	r	**Reserved** Read as 0; must be written with 0.

OTBU1T
OCDS TBU1 Trigger Register (9FDC8$_H$) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
CV	[23:0]	rw	**Compare Value** This value is compared to TBU_CH0_BASE register. As long as both match the associated TS16_IOS.MISC bit (**Table 24-44**) is active.
EN	28	rw	**Enable** 0_B Disabled 1_B Enabled
0	[27:24], [31:29]	r	**Reserved** Read as 0; must be written with 0.

OTBU2T
OCDS TBU2 Trigger Register (9FDCC$_H$) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
CV	[23:0]	rw	**Compare Value** This value is compared to TBU_CH0_BASE register. As long as both match the associated TS16_IOS.MISC bit (**Table 24-44**) is active.
EN	28	rw	**Enable** 0_B Disabled 1_B Enabled
0	[27:24], [31:29]	r	**Reserved** Read as 0; must be written with 0.

24.10 Revision History

Changes from version 0.1

- first version

Changes from version 0.1 to 0.2

- 2.1.1 New Block Diagram
- 2.1.1 TIM trigger out definition
- 2.4 More detailed description of IRQ_NOTIFY clearing
- 2.7 New register GTM_HW_CONF
- 2.8.14 New register GTM_HW_CONF
- 5.1.3 additional input observation functionality
- 5.4.2.5ff remove channel > 0 TBCM limitation
- 5.6 new register TIMi_INP_VAL

- 5.7.1 remove channel > 0 TBCM limitation
  more detailed information about input selection in TGPS and TBCM mode
- 5.7.5 more detailed information about input selection in TBCM mode
- 5.7.7 more detailed information about input selection in TBCM mode
- 5.7.7 register is writable also in TGPS mode
- 5.7.18 additional information about input selection for channels
- 5.7.19 new register TIMi_INP_VAL
- 6.1.1 description of TOM0 channel 0 trigger input added
- 6.2.2 new feature description: additional trigger source is TIM_EXT_CAPTURE
- 6.3.1 update figure
- 6.3.2 update figure
- 6.3.3 update figure
- 6.3ff description of trigger by TIM_EXT_CAPTURE added; behaviour on reset on CN0 by CCU0 compare
- 6.3.6.1ff figure and description of new onw shot trigger feature added
- 6.6.9 new bits OSM_TRIG, EXT_TRIG and EXTRIGOUT added
- 6.6.10 new bits OSM_TRIG, EXT_TRIG and EXTRIGOUT added
- 7.2.2 Definition path delay
- 7.3.1 More detailed description for channel 0 function
- 7.4.1ff fully truth tables
- 7.4.3 Definition pulse generation on edge
- 7.7.1 asynchronous update for 0--
- 7.7.2 No shift function for channel 0
- update mapping to package

Changes from version 0.2 to 1.0

- 2.5 more detailed description of IRQ_NOTIFY
- 2.9.13 bit 0 formula corrected
- 10.2.3 more detailed information about filter reconfiguration
- 10.5 more detailed information about MAP interface
- 10.8.1ff more detailed information about TIM MODE reconfiguration and GELx_SEL
- 10.8.6 additional information about GPR0
- 10.8.7 additional information about GPR1
- 10.8.9 additional information about CNTS
- 10.8.20 additional information about input selection by EXT_CAP_SRC

Changes from version 1.0 to 1.1

- add bit 12 to register GTM_BRIDGE_MODE
- update reset values for registers GTM_REV, GTM_BRIDGE_MODE and GTM_HW_CONF
- update pinning information for QFP100 and bigger packages
- add pinning information for QFP80 package

- removed table Port to GTM DTM Auxillary Mapping
- update figure DTM connections for correct DTM numbering
- update links for register TOUTSELn description
- update description of register DTMx_CTRL bit field CLK_SEL

25 Capture/Compare Unit 6 (CCU6)

The CCU6 is a high-resolution 16-bit capture and compare unit with application specific modes, mainly for AC drive control. Special operating modes support the control of Brushless DC-motors using Hall sensors or Back-EMF detection. Furthermore, block commutation and control mechanisms for multi-phase machines are supported.
It also supports inputs to start several timers synchronously, an important feature in devices with several CCU6 kernels.

This chapter is structured as follows:

- Introduction (see **Section 25.1**)
  including register overview (see **Section 25.1.3**)
- Operating T12 (see **Section 25.2**)
  including T12-related registers (see **Section 25.2.8**)
  and capture/compare control registers (see **Section 25.2.9**)
- Operating T13 (see **Section 25.3**)
  including T13-related registers (see **Section 25.3.6**)
- Synchronous start feature (see **Section 25.4**)
- Trap handling (see **Section 25.5**)
- Multi-Channel mode (see **Section 25.6**)
- Hall sensor mode (see **Section 25.7**)
- Modulation control registers (see **Section 25.8**)
- Interrupt handling (see **Section 25.9**)
  including interrupt registers (see **Section 25.9.2**)
- General module operation (see **Section 25.10**)
  including general registers (see **Section 25.10.5**)
  and BPI registers (see **Section 25.10.6**)
- Module implementation (see **Section 25.11**)

25.1 Introduction

The CCU6 unit is made up of a Timer T12 Block with three capture/compare channels and a Timer T13 Block with one compare channel. The T12 channels can independently generate PWM signals or accept capture triggers, or they can jointly generate control signal patterns to drive AC-motors or inverters.

A rich set of status bits, synchronized updating of parameter values via shadow registers, and flexible generation of interrupt request signals provide means for efficient software-control.

Note: The capture/compare module itself is named CCU6 (capture/compare unit 6).
A capture/compare channel inside this module is named CC6x.

25.1.1 Feature Set Overview

This section gives an overview over the different building blocks and their main features.

Timer 12 Block Features

- Three capture/compare channels, each channel can be used either as capture or as compare channel
- Generation of a three-phase PWM supported (six outputs, individual signals for high-side and low-side switches)
- 16-bit resolution, maximum count frequency = peripheral clock
- Dead-time control for each channel to avoid short-circuits in the power stage
- Concurrent update of T12 registers
- Center-aligned and edge-aligned PWM can be generated
- Single-shot mode supported
- Start can be controlled by external events
- Capability of counting external events
- Many interrupt request sources
- Hysteresis-like control mode

Timer 13 Block Features

- One independent compare channel with one output
- 16-bit resolution, maximum count frequency = peripheral clock
- Concurrent update of T13 registers
- Can be synchronized to T12
- Interrupt generation at period-match and compare-match
- Single-shot mode supported
- Start can be controlled by external events
- Capability of counting external events

Additional Specific Functions

- Block commutation for Brushless DC-drives implemented
- Position detection via Hall-sensor pattern
- Noise filter supported for position input signals
- Automatic rotational speed measurement and commutation control for block commutation
- Integrated error handling
- Fast emergency stop without CPU load via external signal ($\overline{\text{CTRAP}}$)
- Control modes for multi-channel AC-drives
- Output levels can be selected and adapted to the power stage

25.1.2 Block Diagram

The Timer T12 can operate in capture and/or compare mode for its three channels. The modes can also be combined (e.g. a channel operates in compare mode, whereas another channel operates in capture mode). The Timer T13 can operate in compare mode only. The multi-channel control unit generates output patterns which can be modulated by T12 and/or T13. The modulation sources can be selected and combined for the signal modulation.

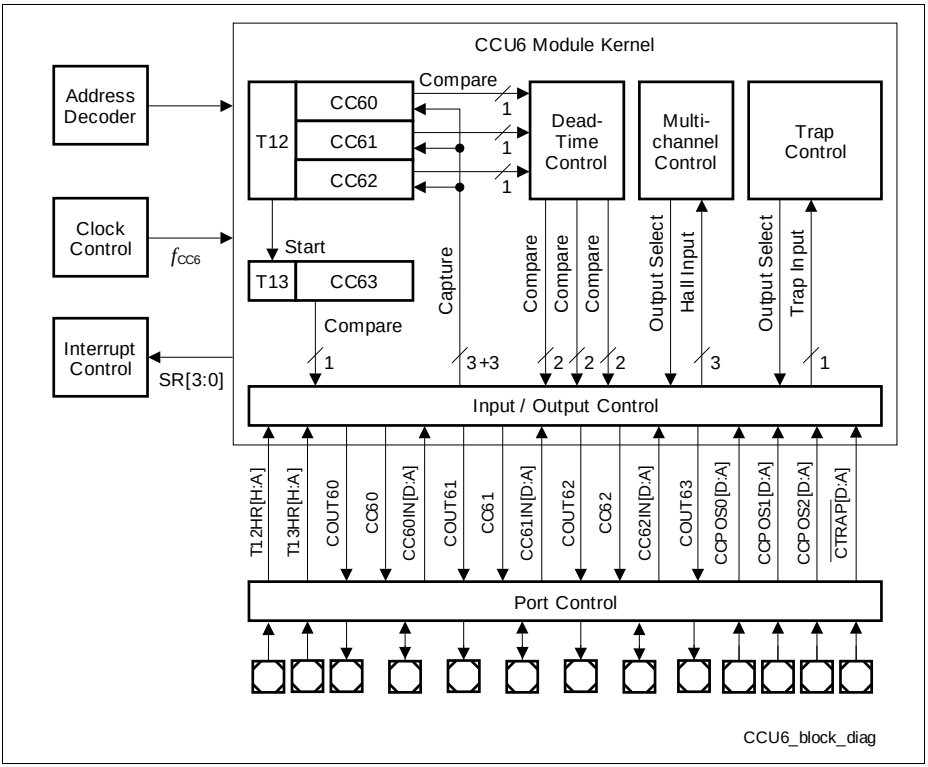


Figure 25-1 CCU6 Block Diagram

25.1.3 CCU6 Kernel Registers

For the generation of the overall register table, the prefix “CCU6x_” has to be added to the register names in this table to identify the registers of different CCU6 kernels that are implemented. In this naming convention, x indicates the kernel number.

Table 25-1 shows all registers required for programming of a CCU6. It summarizes the CCU6 kernel registers and defines their offset addresses. BPI registers are listed separately in **Table 25-14**, and Module Output Select registers are listed in **Table 25-16**.

The CCU6 module has no destructive register read mechanisms.

CCU6 Kernel Register Overview

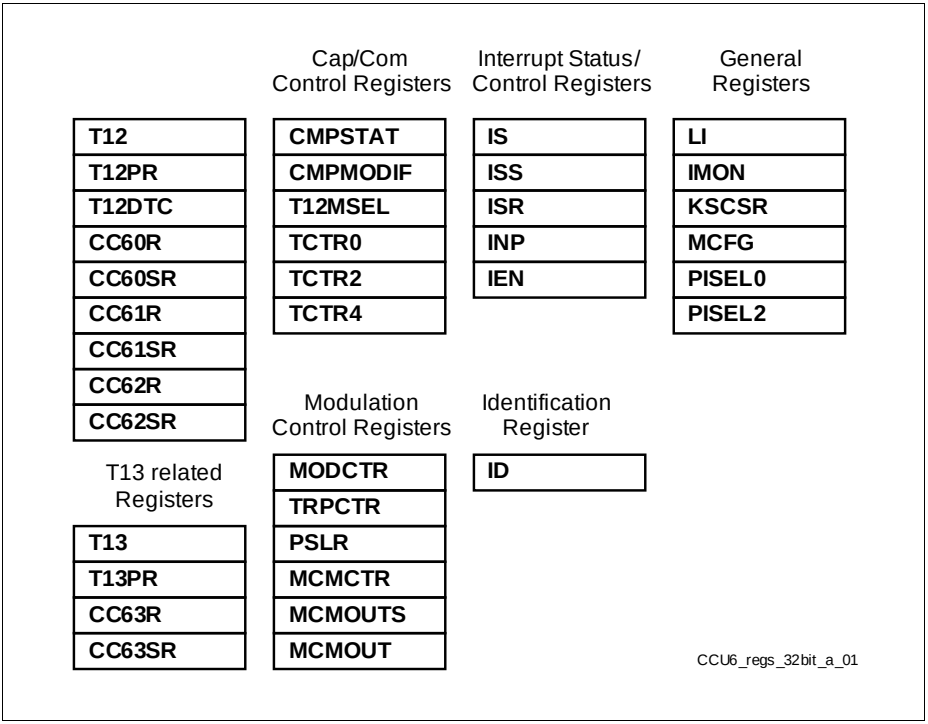


Figure 25-2 CCU6 Registers

Note: In the case of a write access to addresses inside the address range (that is covered by the same chip select signal), but that are not the addresses explicitly mentioned for the module, the write access is not taken into account for the

module. The same principle is valid for read accesses. In case of a read access to another address, the module does not react.

*Note: The exact register address is given by the relative address of the register (given in **Table 25-1**) plus the kernel base address (given in **Table 25-15**) of the kernel.*

Table 25-1 CCU6 Module Registers

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Reset	Page Num.
			Read	Write		
General Registers						
ID	Module Identification Register	08_H	U, SV	BE	Application Reset	25-115
PISEL0	Port Input Select Register 0	10_H	U, SV	U, SV, P	Application Reset	25-116
PISEL2	Port Input Select Register 2	14_H	U, SV	U, SV, P	Application Reset	25-118
KSCSR	Kernel State Control Sensitivity Register	$1C_H$	U, SV	U, SV, P	Debug Reset	25-127
MCFG	Module Configuration Register	04_H	U, SV	U,SV, P	Application Reset	25-121
IMON	Input Monitoring Register	98_H	U, SV	U, SV, P	Application Reset	25-122
LI	Lost Indicator Register	$9C_H$	U, SV	U, SV, P	Application Reset	25-125
Timer T12 related Registers						
T12	Timer 12 Counter Register	20_H	U, SV	U, SV, P	Application Reset	25-33
T12PR	Timer 12 Period Register	24_H	U, SV	U, SV, P	Application Reset	25-34
T12DTC	Dead-Time Control Register for Timer T12	28_H	U, SV	U, SV, P	Application Reset	25-37
CC60R	Capture/Compare Register Channel CC60	30_H	U, SV	U,SV, P	Application Reset	25-35
CC61R	Capture/Compare Register Channel CC61	34_H	U, SV	U,SV, P	Application Reset	25-35

Table 25-1 CCU6 Module Registers (cont'd)

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Reset	Page Num.
			Read	Write		
CC62R	Capture/Compare Register Channel CC62	38_H	U, SV	U,SV, P	Application Reset	25-35
CC60SR	Capture/Compare Shadow Register Channel CC60	40_H	U, SV	U, SV, P	Application Reset	25-36
CC61SR	Capture/Compare Shadow Register Channel CC61	44_H	U, SV	U, SV, P	Application Reset	25-36
CC62SR	Capture/Compare Shadow Register Channel CC62	48_H	U, SV	U, SV, P	Application Reset	25-36
Capture/Compare Control Registers						
CMPSTAT	Compare State Register	60_H	U, SV	U, SV, P	Application Reset	25-39
CMPMODIF	Compare State Modification Register	64_H	U, SV	U, SV, P	Application Reset	25-42
T12MSEL	T12 Capture/Compare Mode Select Register	68_H	U, SV	U, SV, P	Application Reset	25-43
TCTR0	Timer Control Register 0	70_H	U, SV	U, SV, P	Application Reset	25-44
TCTR2	Timer Control Register 2	74_H	U, SV	U, SV, P	Application Reset	25-48
TCTR4	Timer Control Register 4	78_H	U, SV	U, SV, P	Application Reset	25-51
Timer T13 related Registers						
T13	Timer 13 Counter Register	50_H	U, SV	U, SV, P	Application Reset	25-66
T13PR	Timer 13 Period Register	54_H	U, SV	U, SV, P	Application Reset	25-67
CC63R	Compare Register for Timer 13	58_H	U, SV	U,SV, P	Application Reset	25-68

Table 25-1 CCU6 Module Registers (cont'd)

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Reset	Page Num.
			Read	Write		
CC63SR	Compare Shadow Register for Timer 13	$5C_H$	U, SV	U, SV, P	Application Reset	25-69
Modulation Control Registers						
MODCTR	Modulation Control Register	80_H	U, SV	U, SV, P	Application Reset	25-83
TRPCTR	Trap Control Register	84_H	U, SV	U, SV, P	Application Reset	25-85
PSLR	Passive State Level Register	88_H	U, SV	U, SV, P	Application Reset	25-88
MCMOUTS	Multi-Channel Mode Output Shadow Register	$8C_H$	U, SV	U, SV, P	Application Reset	25-92
MCMOUT	Multi-Channel Mode Output Register	90_H	U, SV	U,SV, P	Application Reset	25-93
MCMCTR	Multi-Channel Mode Control Register	94_H	U, SV	U, SV, P	Application Reset	25-89
Interrupt Status and Node Registers						
IS	Interrupt Status Register	$A0_H$	U, SV	U,SV, P	Application Reset	25-98
ISS	Interrupt Status Set Register	$A4_H$	U, SV	U, SV, P	Application Reset	25-101
ISR	Interrupt Status Reset Register	$A8_H$	U, SV	U, SV, P	Application Reset	25-103
INP	Interrupt Node Pointer Register	AC_H	U, SV	U, SV, P	Application Reset	25-108
IEN	Interrupt Node Pointer Register	$B0_H$	U, SV	U, SV, P	Application Reset	25-105

25.2 Operating Timer T12

The timer T12 block is the main unit to generate the 3-phase PWM signals. A 16-bit counter is connected to 3 channel registers via comparators, that generate a signal when the counter contents match one of the channel register contents. A variety of control functions facilitate the adaptation of the T12 structure to different application needs.
Besides the 3-phase PWM generation, the T12 block offers options for individual compare and capture functions, as well as dead-time control and hysteresis-like compare mode.

This section provides information about:

- T12 overview (see **Section 25.2.1**)
- Counting scheme (see **Section 25.2.2**)
- Compare modes (see **Section 25.2.3**)
- Compare mode output path (see **Section 25.2.4**)
- Capture modes (see **Section 25.2.5**)
- Shadow transfer (see **Section 25.2.6**)
- T12 operating mode selection (see **Section 25.2.7**)
- T12 register description (see **Section 25.2.8**)

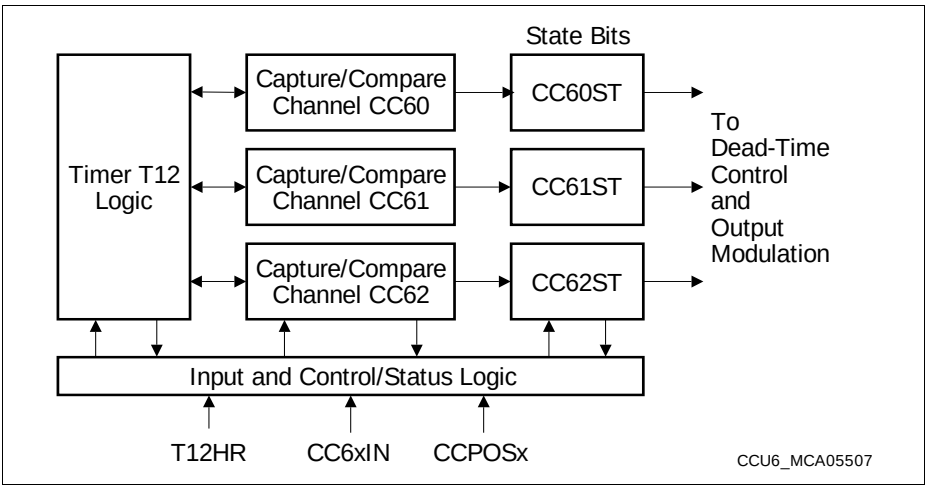


Figure 25-3 Overview Diagram of the Timer T12 Block

25.2.1 T12 Overview

Figure 25-4 shows a detailed block diagram of Timer T12. The functions of the timer T12 block are controlled by bits in registers **TCTR0**, **TCTR2**, and **PISEL0**.

Timer T12 receives its input clock (f_{T12}) from the module clock f_{CC6} via a programmable prescaler and an optional 1/256 divider or from an input signal T12HR. These options are controlled via bit fields T12CLK and T12PRE (see **Table 25-2**). T12 can count up or down, depending on the selected operation mode. A direction flag, CDIR, indicates the current counting direction.

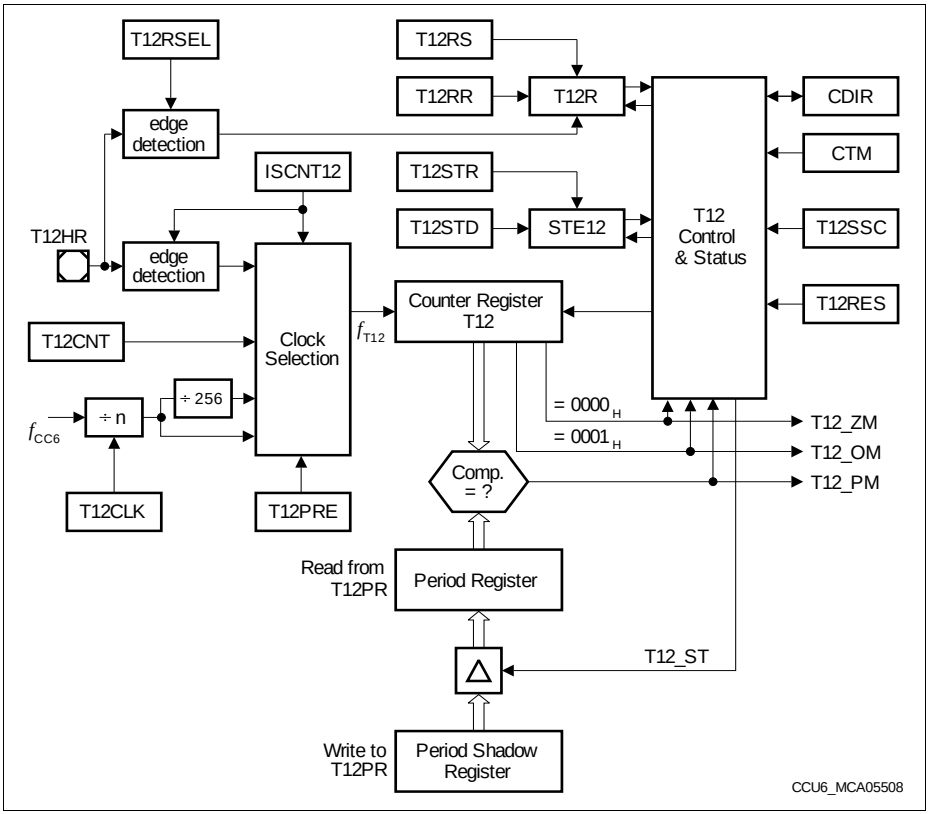


Figure 25-4 Timer T12 Logic and Period Comparators

Via a comparator, the T12 counter register **T12** is connected to a Period Register **T12PR**. This register determines the maximum count value for T12.

In Edge-Aligned mode, T12 is cleared to 0000_H after it has reached the period value defined by T12PR. In Center-Aligned mode, the count direction of T12 is set from ‘up’ to

‘down’ after it has reached the period value (please note that in this mode, T12 exceeds the period value by one before counting down). In both cases, signal T12_PM (T12 Period Match) is generated. The Period Register receives a new period value from its Shadow Period Register.
A read access to T12PR delivers the current period value at the comparator, whereas a write access targets the Shadow Period Register to prepare another period value. The transfer of a new period value from the Shadow Period Register into the Period Register (see **Section 25.2.6**) is controlled via the ‘T12 Shadow Transfer’ control signal, T12_ST. The generation of this signal depends on the operating mode and on the shadow transfer enable bit STE12. Providing a shadow register for the period value as well as for other values related to the generation of the PWM signal allows a concurrent update by software for all relevant parameters.
Two further signals indicate whether the counter contents are equal to 0000_H (T12_ZM = zero match) or 0001_H (T12_OM = one match). These signals control the counting and switching behavior of T12.

The basic operating mode of T12, either Edge-Aligned mode (**Figure 25-5**) or Center-Aligned mode (**Figure 25-6**), is selected via bit CTM. A Single-Shot control bit, T12SSC, enables an automatic stop of the timer when the current counting period is finished (see **Figure 25-7** and **Figure 25-8**).

The start or stop of T12 is controlled by the Run bit T12R that can be modified by bits in register **TCTR4**. The run bit can be set/cleared by software via the associated set/clear bits T12RS or T12RR, it can be set by a selectable edge of the input signal T12HR (**TCTR2**.T12RSEL), or it is cleared by hardware according to preselected conditions.
The timer T12 run bit T12R must not be set while the applied T12 period value is zero.
Timer T12 can be cleared via control bit T12RES. Setting this write-only bit does only clear the timer contents, but has no further effects, for example, it does not stop the timer.

The generation of the T12 shadow transfer control signal, T12_ST, is enabled via bit STE12. This bit can be set or reset by software indirectly through its associated set/clear control bits T12STR and T12STD.
While Timer T12 is running, write accesses to the count register T12 are not taken into account. If T12 is stopped and the Dead-Time counters are 0, write actions to register T12 are immediately taken into account.

25.2.2 T12 Counting Scheme

This section describes the clocking and counting capabilities of T12.

25.2.2.1 Clock Selection

In **Timer Mode** (**PISEL2**.ISCNT12 = 00_B), the input clock f_{T12} of Timer T12 is derived from the internal module clock f_{CC6} through a programmable prescaler and an optional 1/256 divider. The resulting prescaler factors are listed in **Table 25-2**. The prescaler of T12 is cleared while T12 is not running (**TCTR0**.T12R = 0) to ensure reproducible timings and delays.

Table 25-2 Timer T12 Input Frequency Options

T12CLK	Resulting Input Clock f_{T12} Prescaler Off (T12PRE = 0)	Resulting Input Clock f_{T12} Prescaler On (T12PRE = 1)
000_B	f_{CC6}	f_{CC6} / 256
001_B	f_{CC6} / 2	f_{CC6} / 512
010_B	f_{CC6} / 4	f_{CC6} / 1024
011_B	f_{CC6} / 8	f_{CC6} / 2048
100_B	f_{CC6} / 16	f_{CC6} / 4096
101_B	f_{CC6} / 32	f_{CC6} / 8192
110_B	f_{CC6} / 64	f_{CC6} / 16384
111_B	f_{CC6} / 128	f_{CC6} / 32768

In **Counter Mode**, timer T12 counts one step:

- If a 1 is written to **TCTR4**.T12CNT and **PISEL2**.ISCNT12 = 01_B
- If a rising edge of input signal T12HR is detected and **PISEL2**.ISCNT12 = 10_B
- If a falling edge of input signal T12HR is detected and **PISEL2**.ISCNT12 = 11_B

25.2.2.2 Edge-Aligned / Center-Aligned Mode

In **Edge-Aligned Mode** (CTM = 0), timer T12 is always counting upwards (CDIR = 0). When reaching the value given by the period register (period-match T12_PM), the value of T12 is cleared with the next counting step (saw tooth shape).

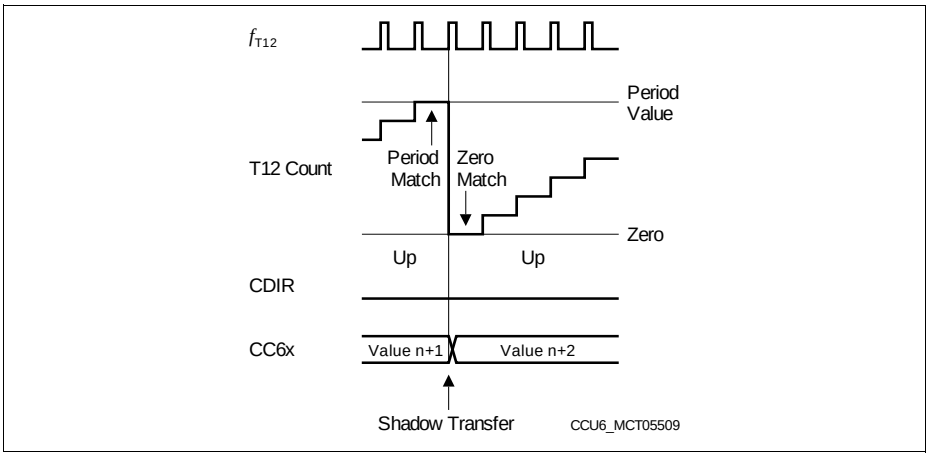


Figure 25-5 T12 Operation in Edge-Aligned Mode

As a result, in Edge-Aligned mode, the timer period is given by:

$T12_{PER}$ = <Period-Value> + 1; in T12 clocks (f_{T12}) (25.1)

In **Center-Aligned Mode** (CTM = 1), timer T12 is counting upwards or downwards (triangular shape). When reaching the value given by the period register (period-match T12_PM) while counting upwards (CDIR = 0), the counting direction control bit CDIR is changed to downwards (CDIR = 1) with the next counting step.
When reaching the value 0001_H (one-match T12_OM) while counting downwards, the counting direction control bit CDIR is changed to upwards with the next counting step.

As a result, in Center.Aligned mode, the timer period is given by:

$T12_{PER}$ = (<Period-Value> + 1) x 2; in T12 clocks (f_{T12}) (25.2)

- With the next clock event of f_{T12} the count direction is set to counting up (CDIR = 0) when the counter reaches 0001_H while counting down.
- With the next clock event of f_{T12} the count direction is set to counting down (CDIR = 1) when the Period-Match is detected while counting up.
- With the next clock event of f_{T12} the counter counts up while CDIR = 0 and it counts down while CDIR = 1.

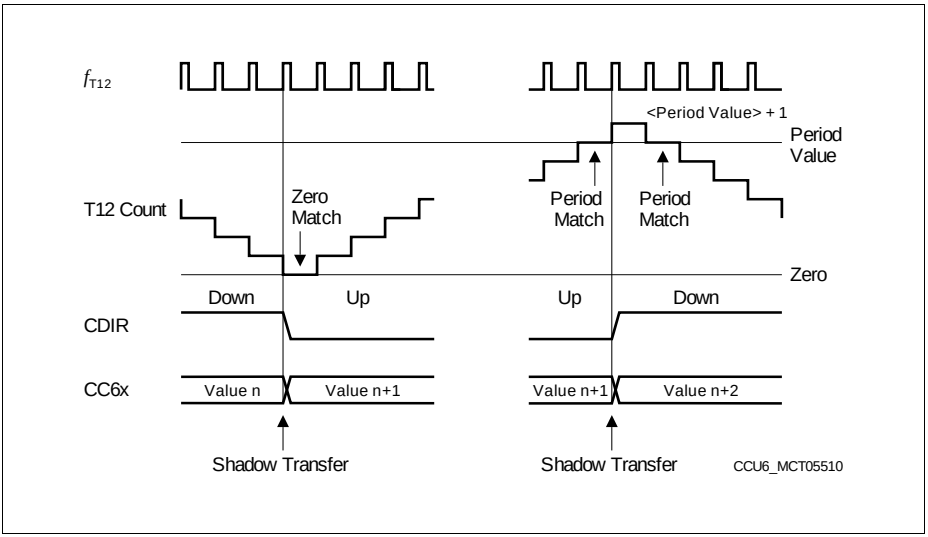


Figure 25-6 T12 Operation in Center-Aligned Mode

*Note: Bit CDIR changes with the next timer clock event after the one-match or the period-match. Therefore, the timer continues counting in the previous direction for one cycle before actually changing its direction (see **Figure 25-6**).*

25.2.2.3 Single-Shot Mode

In Single-Shot Mode, the timer run bit T12R is cleared by hardware. If bit T12SSC = 1, the timer T12 will stop when the current timer period is finished.

In Edge-Aligned mode, T12R is cleared when the timer becomes zero after having reached the period value (see **Figure 25-7**).

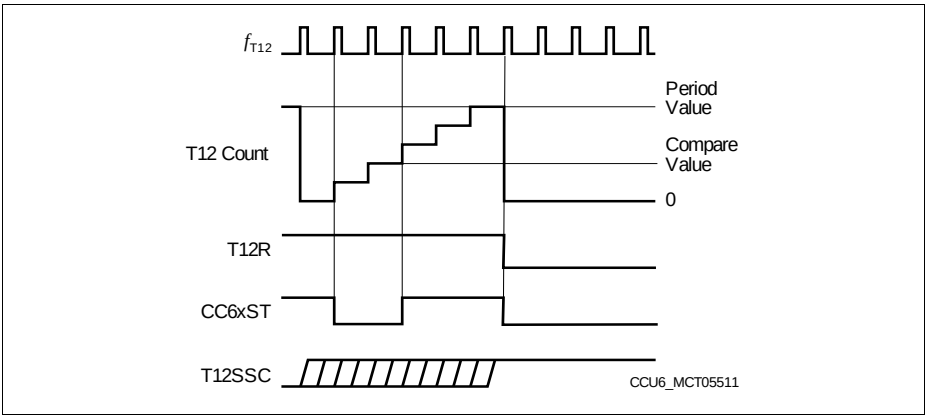


Figure 25-7 Single-Shot Operation in Edge-Aligned Mode

In Center-Aligned mode, the period is finished when the timer has counted down to zero (one clock cycle after the one-match while counting down, see **Figure 25-8**).

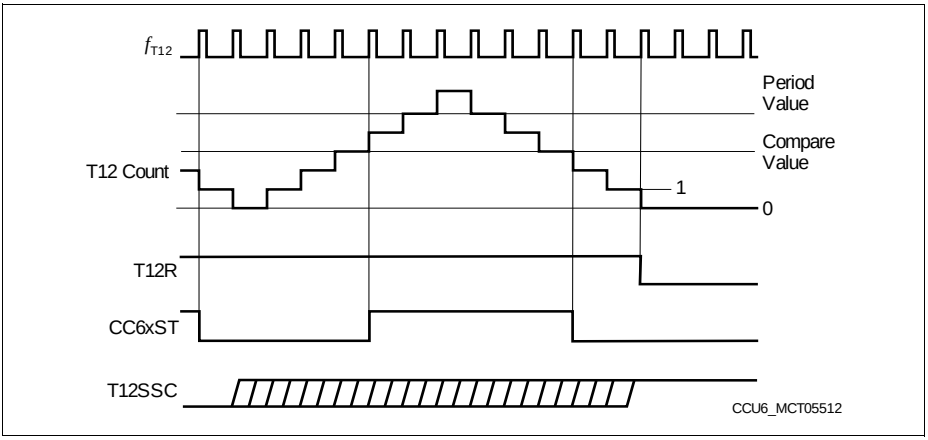


Figure 25-8 Single-Shot Operation in Center-Aligned Mode

25.2.3 T12 Compare Mode

Associated with Timer T12 are three individual capture/compare channels, that can perform compare or capture operations with regard to the contents of the T12 counter. The capture functions are explained in **Section 25.2.5**.

25.2.3.1 Compare Channels

In Compare Mode (see **Figure 25-9**), the three individual compare channels CC60 CC61, and CC62 can generate a three-phase PWM pattern.

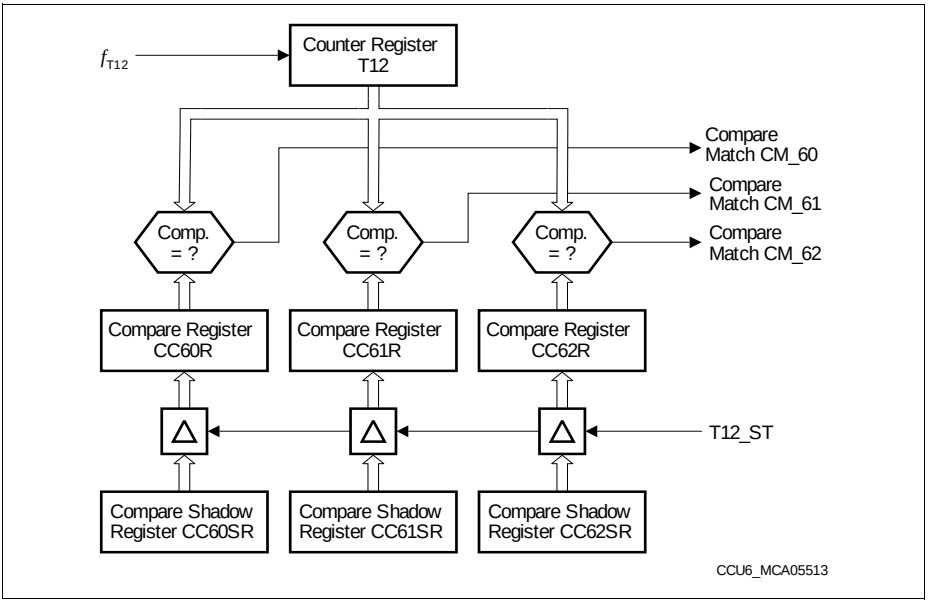


Figure 25-9 T12 Channel Comparators

Each compare channel is connected to the T12 counter register via its individual equal-to comparator, generating a match signal when the contents of the counter matches the contents of the associated compare register. Each channel consists of the comparator and a double register structure - the actual compare register CC6xR, feeding the comparator, and an associated shadow register CC6xSR, that is preloaded by software and transferred into the compare register when signal T12 shadow transfer, T12_ST, gets active. Providing a shadow register for the compare value as well as for other values related to the generation of the PWM signal facilitates a concurrent update by software for all relevant parameters of a three-phase PWM.

25.2.3.2 Channel State Bits

Associated with each (compare) channel is a State Bit, **CMPSTAT**.CC6xST, holding the status of the compare (or capture) operation (see **Figure 25-10**). In compare mode, the State Bits are modified according to a set of switching rules, depending on the current status of timer T12.

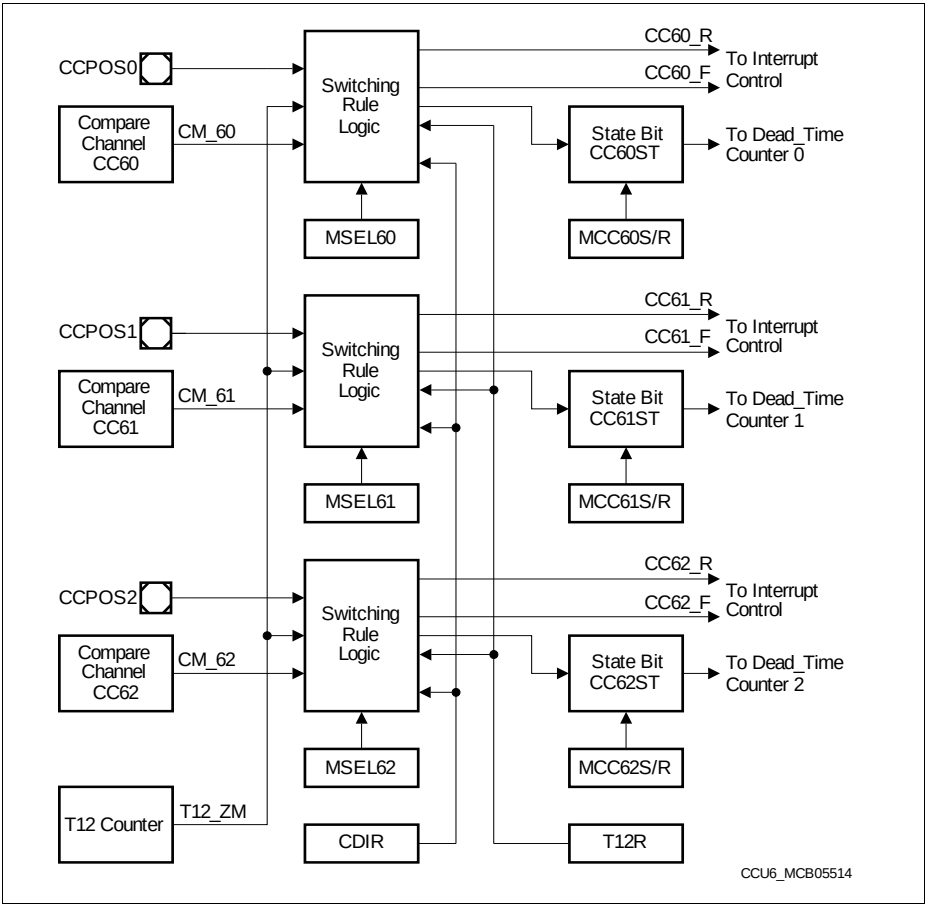


Figure 25-10 Compare State Bits for Compare Mode

The inputs to the switching rule logic for the CC6xST bits are the timer direction (CDIR), the timer run bit (T12R), the timer T12 zero-match signal (T12_ZM), and the actual individual compare-match signals CM_6x as well as the mode control bits, **T12MSEL**.MSEL6x.

In addition, each state bit can be set or cleared by software via the appropriate set and reset bits in register **CMPMODIF**, MCC6xS and MCC6xR. The input signals CCPOSx are used in hysteresis-like compare mode, whereas in normal compare mode, these inputs are ignored.

Note: In Hall Sensor, single shot or capture modes, additional/different rules are taken into account (see related sections).

A compare interrupt event CC6x_R is signaled when a compare match is detected while counting upwards, whereas the compare interrupt event CC6x_F is signaled when a compare match is detected while counting down. The actual setting of a State Bit has no influence on the interrupt generation in compare mode.

A modification of a State Bit CC6xST by the switching rule logic due to a compare action is only possible while Timer T12 is running (T12R = 1). If this is the case, the following switching rules apply for setting and clearing the State Bits in Compare Mode (illustrated in **Figure 25-11** and **Figure 25-12**):

A State Bit **CC6xST is set** to 1:

- with the next T12 clock (f_{T12}) after a compare-match when T12 is counting up (i.e., when the counter is incremented above the compare value);
- with the next T12 clock (f_{T12}) after a zero-match AND a parallel compare-match when T12 is counting up.

A State Bit **CC6xST is cleared** to 0:

- with the next T12 clock (f_{T12}) after a compare-match when T12 is counting down (i.e., when the counter is decremented below the compare value in center-aligned mode);
- with the next T12 clock (f_{T12}) after a zero-match AND NO parallel compare-match when T12 is counting up.

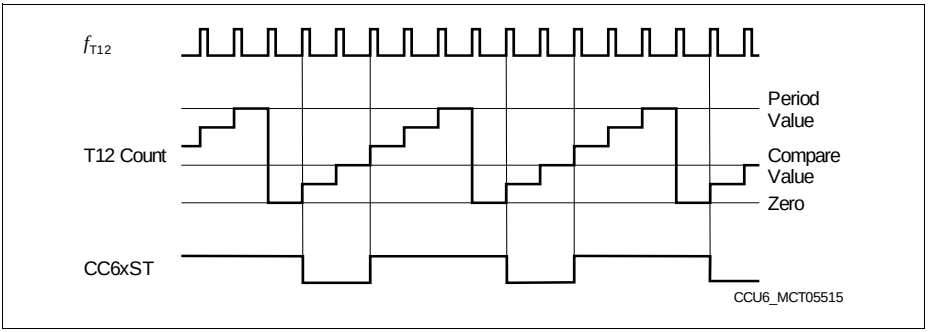


Figure 25-11 Compare Operation, Edge-Aligned Mode

Figure 25-13 illustrates some more examples for compare waveforms. It is important to note that in these examples, it is assumed that some of the compare values are changed

while the timer is running. This change is performed via a software preload of the Shadow Register, CC6xSR. The value is transferred to the actual Compare Register CC6xR with the T12 Shadow Transfer signal, T12_ST, that is assumed to be enabled.

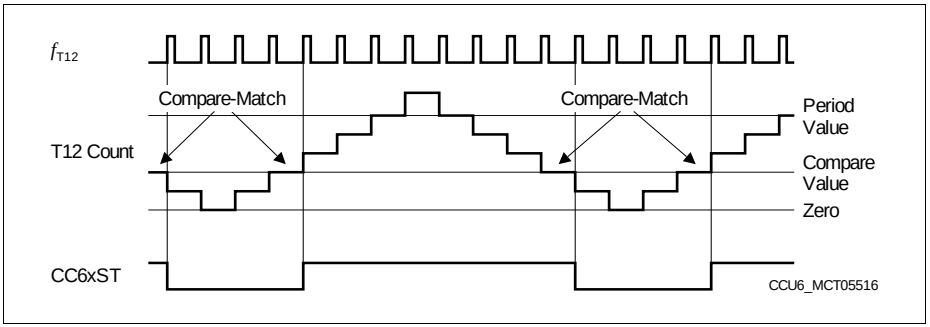


Figure 25-12 Compare Operation, Center-Aligned Mode

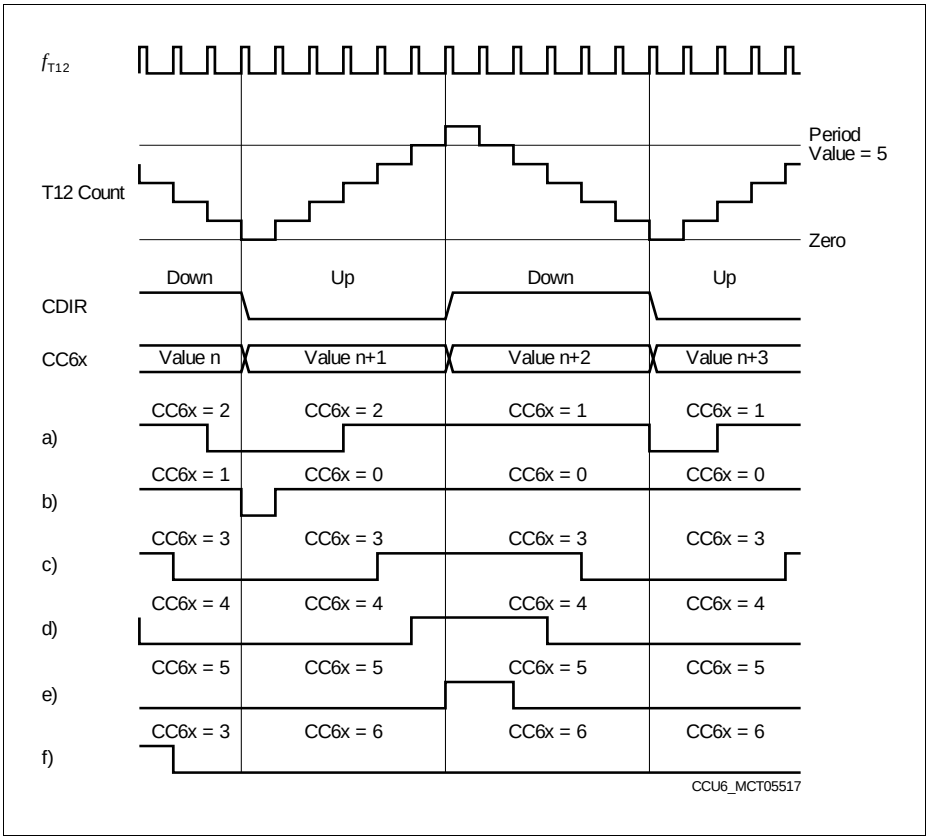


Figure 25-13 Compare Waveform Examples

Example b) illustrates the transition to a duty cycle of 100%. First, a compare value of 0001_H is used, then changed to 0000_H. Please note that a low pulse with the length of one T12 clock is still produced in the cycle where the new value 0000_H is in effect; this pulse originates from the previous value 0001_H. In the following timer cycles, the State Bit CC6xST remains at 1, producing a 100% duty cycle signal. In this case, the compare rule ‘zero-match AND compare-match’ is in effect.

Example f) shows the transition to a duty cycle of 0%. The new compare value is set to <Period-Value> + 1, and the State Bit CC6ST remains cleared.

Figure 25-14 illustrates an example for the waveforms of all three channels. With the appropriate dead-time control and output modulation, a very efficient 3-phase PWM signal can be generated.

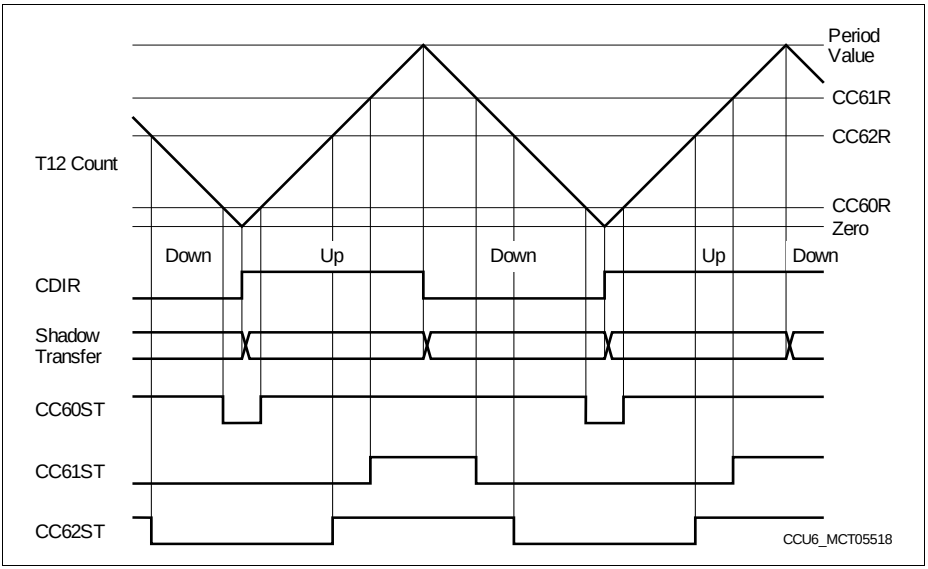


Figure 25-14 Three-Channel Compare Waveforms

25.2.3.3 Hysteresis-Like Control Mode

The hysteresis-like control mode (**T12MSEL**.MSEL6x = 1001_B) offers the possibility to switch off the PWM output if the input CCPOSx becomes 0 by clearing the State Bit CC6xST. This can be used as a simple motor control feature by using a comparator indicating, e.g., overcurrent. While CCPOSx = 0, the PWM outputs of the corresponding channel are driving their passive levels, because the setting of bit CC6xST is only possible while CCPOSx = 1.
As long as input CCPOSx is 0, the corresponding State Bit is held 0. When CCPOSx is at high level, the outputs can be in active state and are determined by bit CC6xST (see **Figure 25-10** for the state bit logic and **Figure 25-15** for the output paths).
The CCPOSx inputs are evaluated with f_{CC6}.

This mode can be used to introduce a timing-related behavior to a hysteresis controller. A standard hysteresis controller detects if a value exceeds a limit and switches its output according to the compare result. Depending on the operating conditions, the switching frequency and the duty cycle are not fixed, but change permanently.

If (outer) time-related control loops based on a hysteresis controller in an inner loop should be implemented, the outer loops show a better behavior if they are synchronized to the inner loops. Therefore, the hysteresis-like mode can be used, that combines timer-related switching with a hysteresis controller behavior. For example, in this mode, an output can be switched on according to a fixed time base, but it is switched off as soon as a falling edge is detected at input CCPOSx.

This mode can also be used for standard PWM with overcurrent protection. As long as there is no low level signal at pin CCPOSx, the output signals are generated in the normal manner as described in the previous sections. Only if input CCPOSx shows a low level, e.g. due to the detection of overcurrent, the outputs are shut off to avoid harmful stress to the system.

25.2.4 Compare Mode Output Path

Figure 25-15 gives an overview on the signal path from a channel State Bit to its output pin in its simplest form. As illustrated, a user has a variety of controls to determine the desired output signal switching behavior in relation to the current state of the State Bit, CC6xST. Please refer to **Section 25.2.4.3** for details on the output modulation.

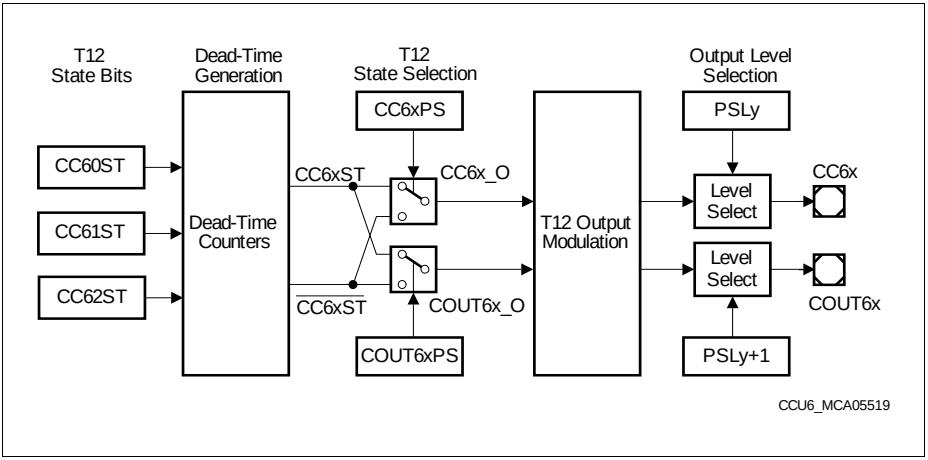


Figure 25-15 Compare Mode Simplified Output Path Diagram

The output path is based on signals that are defined as active or passive. The terms active and passive are not related to output levels, but to internal actions. This mainly applies for the modulation, where T12 and T13 signals are combined with the multi-channel signals and the trap function. The Output level Selection allows the user to define the output level at the output pin for the passive state (inverted level for the active state). It is recommended to configure this block in a way that an external power switch is switched off while the CCU6 delivers an output signal in the passive state.

25.2.4.1 Dead-Time Generation

The generation of (complementary) signals for the high-side and the low-side switches of one power inverter phase is based on the same compare channel. For example, if the high-side switch should be active while the T12 counter value is above the compare value (State Bit = 1), then the low-side switch should be active while the counter value is below the compare value (State Bit = 0).

In most cases, the switching behavior of the connected power switches is not symmetrical concerning the switch-on and switch-off times. A general problem arises if the time for switch-on is smaller than the time for switch-off of the power device. In this case, a short-circuit can occur in the inverter bridge leg, which may damage the complete system. In order to solve this problem by HW, this capture/compare unit

contains a programmable Dead-Time Generation Block, that delays the passive to active edge of the switching signals by a programmable time (the active to passive edge is not delayed).

The Dead-Time Generation Block, illustrated in **Figure 25-16**, is built in a similar way for all three channels of T12. It is controlled by bits in register **T12DTC**. Any change of a CC6xST State Bit activates the corresponding Dead-Time Counter, that is clocked with the same input clock as T12 (f_{T12}). The length of the dead-time can be programmed by bit field DTM. This value is identical for all three channels. Writing **TCTR4**.DTRES = 1 sets all dead-times to passive.

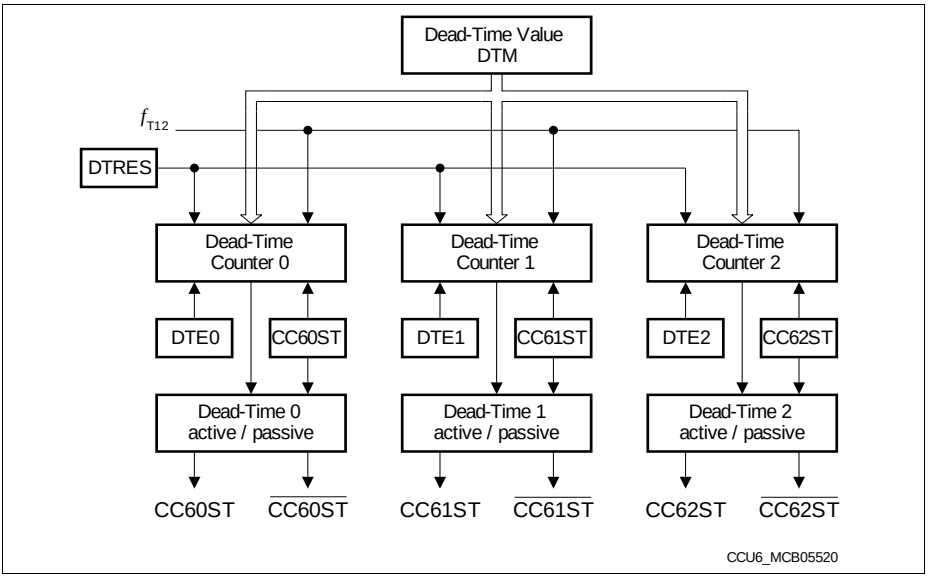


Figure 25-16 Dead-Time Generation Block Diagram

Each of the three dead-time counters has its individual dead-time enable bit, DTEx. An enabled dead-time counter generates a dead-time delaying the passive-to-active edge of the channel output signal. The change in a State Bit CC6xST is not taken into account while the dead-time generation of this channel is currently in progress (active). This avoids an unintentional additional dead-time if a State Bit CC6xST changes too early.
A disabled dead-time counter is always considered as passive and does not delay any edge of CC6xST.
Based on the State Bits CC6xST, the Dead-Time Generation Block outputs a direct signal CC6xST and an inverted signal $\overline{CC6xST}$ for each compare channel, each masked with the effect of the related Dead-Time Counters (waveforms illustrated in **Figure 25-17**).

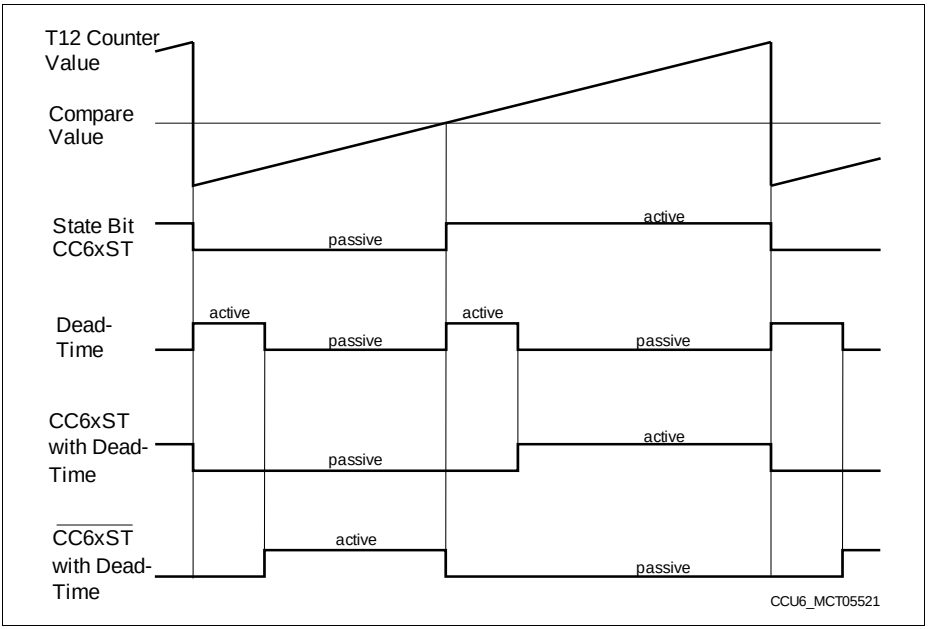


Figure 25-17 Dead-Time Generation Waveforms

25.2.4.2 State Selection

To support a wide range of power switches and drivers, the state selection offers the flexibility to define when an output can be active and can be modulated, especially useful for **complementary or multi-phase PWM** signals.

The state selection is based on the signals CC6xST and $\overline{\text{CC6xST}}$ delivered by the dead-time generator (see **Figure 25-15**). Both signals are never active at the same time, but can be passive at the same time. This happens during the dead-time of each compare channel after a change of the corresponding State Bit CC6xST.

The user can select independently for each output signal CC6xO and COUT6xO if it should be active before or after the compare value has been reached (see register **CMPSTAT**). With this selection, the active (conducting) phases of complementary power switches in a power inverter bridge leg can be positioned with respect to the compare value (e.g. signal CC6xO can be active before, whereas COUT6xO can be active after the compare value is reached). Like this, the output modulation, the trap logic and the output level selection can be programmed independently for each output signal, although two output signals are referring to the same compare channel.

25.2.4.3 Output Modulation and Level Selection

The last block of the data path is the Output Modulation block. Here, all the modulation sources and the trap functionality are combined and control the actual level of the output pins (controlled by the modulation enable bits T1xMODENy and MCMEN in register **MODCTR**). The following signal sources can be combined here **for each T12 output signal** (see **Figure 25-18** for compare channel CC60):

- A **T12 related compare signal** CC6x_O (for outputs CC6x) or COUT6x_O (for outputs COUT6x) delivered by the T12 block (state selection with dead-time) with an individual enable bit T12MODENy per output signal (y = 0, 2, 4 for outputs CC6x and y = 1, 3, 5 for outputs COUT6x)
- The **T13 related compare signal** CC63_O delivered by the T13 state selection with an individual enable bit T13MODENy per output signal (y = 0, 2, 4 for outputs CC6x and y = 1, 3, 5 for outputs COUT6x)
- A **multi-channel output signal** MCMPy (y = 0, 2, 4 for outputs CC6x and y = 1, 3, 5 for outputs COUT6x) with a common enable bit MCMEN
- The **trap state** TRPS with an individual enable bit TRPENy per output signal (y = 0, 2, 4 for outputs CC6x and y = 1, 3, 5 for outputs COUT6x)

If one of the modulation input signals CC6x_O/COUT6x_O, CC63_O, or MCMPy of an output modulation block is enabled and is at passive state, the modulated is also in passive state, regardless of the state of the other signals that are enabled. Only if all enabled signals are in active state the modulated output shows an active state. If no modulation input is enabled, the output is in passive state.
If the Trap State is active (TRPS = 1), then the outputs that are enabled for the trap signal (by TRPENy = 1) are set to the passive state.

The output of each of the modulation control blocks is connected to a level select block that is configured by register **PSLR**. It offers the option to determine the actual output level of a pin, depending on the state of the output line (decoupling of active/passive state and output polarity) as specified by the Passive State Select bit PSLy. If the modulated output signal is in the passive state, the level specified directly by PSLy is output. If it is in the active state, the inverted level of PSLy is output. This allows the user to adapt the polarity of an active output signal to the connected circuitry.

The PSLy bits have shadow registers to allow for updates without undesired pulses on the output lines. The bits related to CC6x and COUT6x (x = 0, 1, 2) are updated with the T12 shadow transfer signal (T12_ST). A read action returns the actually used values, whereas a write action targets the shadow bits. Providing a shadow register for the PSL value as well as for other values related to the generation of the PWM signal facilitates a concurrent update by software for all relevant parameters.

Figure 25-18 shows the output modulation structure for compare channel CC60 (output signals CC60 and COUT60). A similar structure is implemented for the other two compare channels CC61 and CC62.

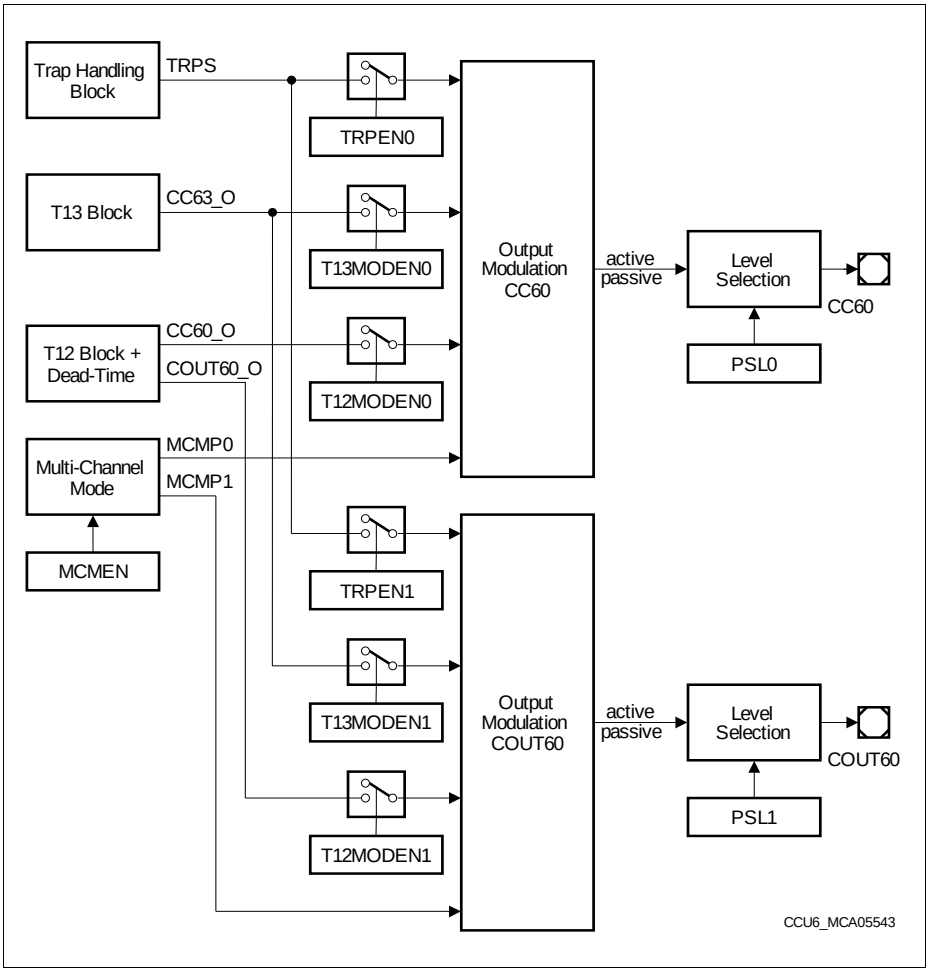


Figure 25-18 Output Modulation for Compare Channel CC60

25.2.5 T12 Capture Modes

Each of the three channels of the T12 Block can also be used to capture T12 time information in response to an external signal CC6xIN.
In capture mode, the interrupt event CC6x_R is detected when a rising edge is detected at the input CC6xIN, whereas the interrupt event CC6x_F is detected when a falling edge is detected.

There are a number of different modes for capture operation. In all modes, both of the registers of a channel are used. The selection of the capture modes is done via the **T12MSEL**.MSEL6x bit fields and can be selected individually for each of the channels.

Table 25-3 Capture Modes Overview

MSEL6x	Mode	Signal	Active Edge	CC6nSR Stored in	T12 Stored in
0100_B	1	CC6xIN	Rising	–	CC6xR
		CC6xIN	Falling	–	CC6xSR
0101_B	2	CC6xIN	Rising	CC6xR	CC6xSR
0110_B	3	CC6xIN	Falling	CC6xR	CC6xSR
0111_B	4	CC6xIN	Any	CC6xR	CC6xSR

Figure 25-19 illustrates **Capture Mode 1**. When a rising edge (0-to-1 transition) is detected at the corresponding input signal CC6xIN, the current contents of Timer T12 are captured into register CC6xR. When a falling edge (1-to-0 transition) is detected at the input signal CC6xIN, the contents of Timer T12 are captured into register CC6xSR.

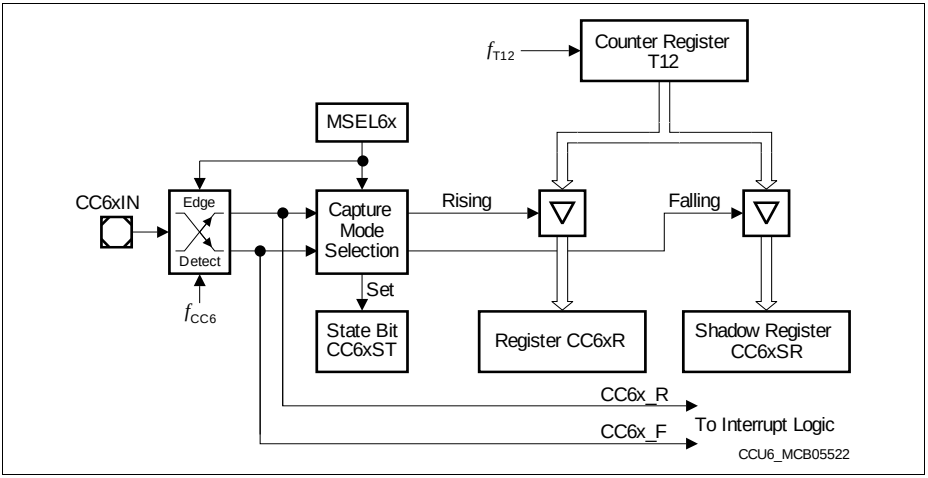


Figure 25-19 Capture Mode 1 Block Diagram

Capture Modes 2, 3 and 4 are shown in **Figure 25-20**. They differ only in the active edge causing the capture operation. In each of the three modes, when the selected edge is detected at the corresponding input signal CC6xIN, the current contents of the shadow register CC6xSR are transferred into register CC6xR, and the current Timer T12 contents are captured in register CC6xSR (simultaneous transfer). The active edge is a rising edge of CC6xIN for Capture Mode 2, a falling edge for Mode 3, and both, a rising or a falling edge for Capture Mode 4, as shown in **Table 25-3**. These capture modes are very useful in cases where there is little time between two consecutive edges of the input signal.

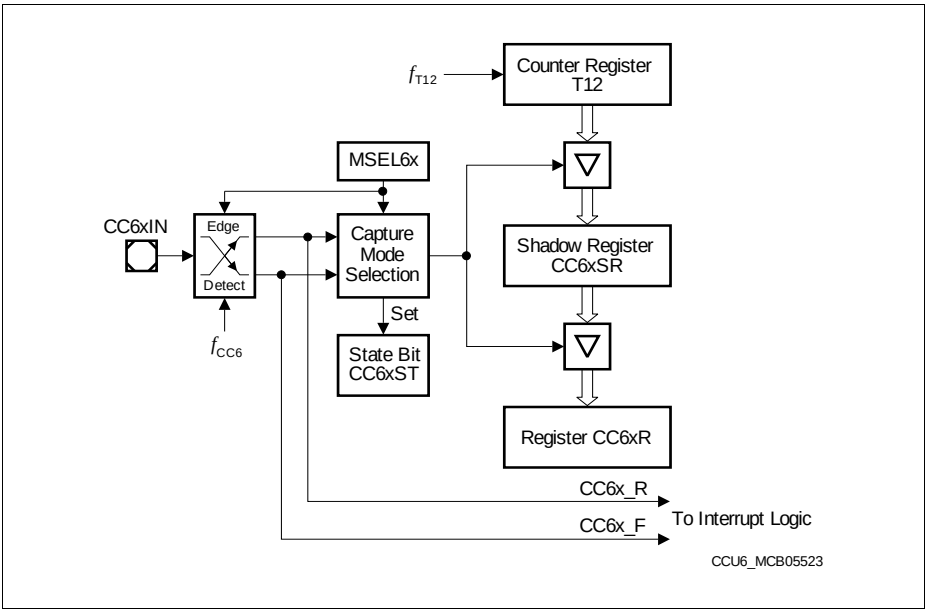


Figure 25-20 Capture Modes 2, 3 and 4 Block Diagram

Five further capture modes are called **Multi-Input Capture Modes**, as they use two different external inputs, signal CC6xIN and signal CCPOSx.

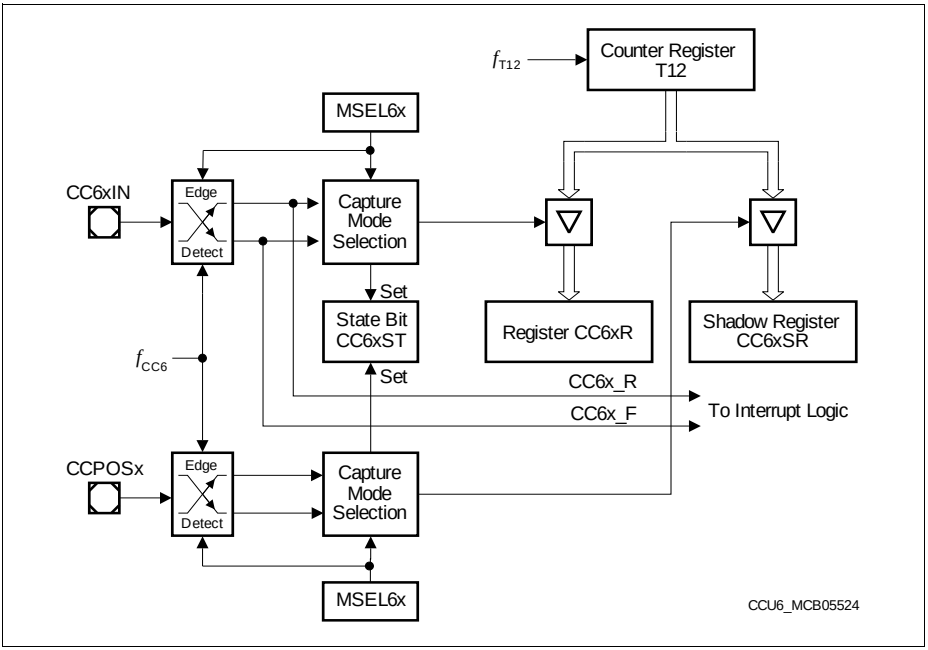


Figure 25-21 Multi-Input Capture Modes Block Diagram

In each of these modes, the current T12 contents are captured in register CC6xR in response to a selected event at signal CC6xIN, and in register CC6xSR in response to a selected event at signal CCPOSx. The possible events can be opposite input transitions, or the same transitions, or any transition at the two inputs. The different options are detailed in **Table 25-4**.

In each of the various capture modes, the Channel State Bit, CC6xST, is set to 1 when the selected capture trigger event at signal CC6xIN or CCPOSx has occurred. The State Bit is not cleared by hardware, but can be cleared by software.

In addition, appropriate signal lines to the interrupt logic are activated, that can generate an interrupt request to the CPU. Regardless of the selected active edge, all edges detected at signal CC6xIN can lead to the activation of the appropriate interrupt request line (see also **Section 25.9**).

Table 25-4 Multi-Input Capture Modes Overview

MSEL6x	Mode	Signal	Active Edge	T12 Stored in
1010_B	5	CC6xIN	Rising	CC6xR
		CCPOSx	Falling	CC6xSR
1011_B	6	CC6xIN	Falling	CC6xR
		CCPOSx	Rising	CC6xSR
1100_B	7	CC6xIN	Rising	CC6xR
		CCPOSx	Rising	CC6xSR
1101_B	8	CC6xIN	Falling	CC6xR
		CCPOSx	Falling	CC6xSR
1110_B	9	CC6xIN	Any	CC6xR
		CCPOSx	Any	CC6xSR
1111_B	–	reserved (no capture or compare action)		

25.2.6 T12 Shadow Register Transfer

A special shadow transfer signal (T12_ST) can be generated to facilitate updating the period and compare values of the compare channels CC60, CC61, and CC62 synchronously to the operation of T12. Providing a shadow register for values defining one PWM period facilitates a concurrent update by software for all relevant parameters. The next PWM period can run with a new set of parameters. The generation of this signal is requested by software via bit **TCTR0**.STE12 (set by writing 1 to the write-only bit **TCTR4**.T12STR, cleared by writing 1 to the write-only bit **TCTR4**.T12STD).

Figure 25-22 shows the shadow register structure and the shadow transfer signals, as well as on the read/write accessibility of the various registers.

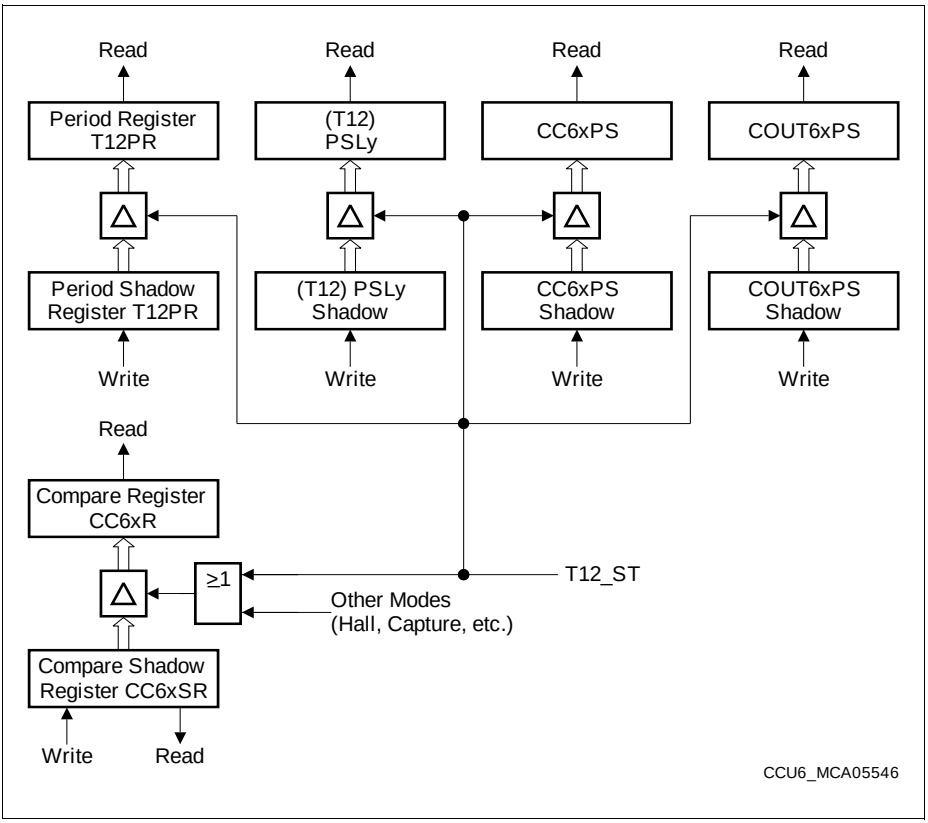


Figure 25-22 T12 Shadow Register Overview

A T12 shadow register transfer takes place (T12_ST active):

- while timer T12 is not running (T12R = 0), or
- STE12 = 1 and a Period-Match is detected while counting up, or
- STE12 = 1 and a One-Match is detected while counting down

When signal T12_ST is active, a shadow register transfer is triggered with the next cycle of the T12 clock. Bit STE12 is automatically cleared with the shadow register transfer.

25.2.7 Timer T12 Operating Mode Selection

The operating mode for the T12 channels are defined by the bit fields **T12MSEL**.MSEL6x.

Table 25-5 T12 Capture/Compare Modes Overview

MSEL6x	Selected Operating Mode
0000_B, 1111_B	Capture/Compare modes switched off
0001_B, 0010_B, 0011_B	Compare mode, see **Section 25.2.3** same behavior for all three codings
$01XX_B$	Double-Register Capture modes, see **Section 25.2.5**
1000_B	Hall Sensor Mode, see **Section 25.7** In order to properly enable this mode, all three MSEL6x fields have to be programmed to Hall Sensor mode.
1001_B	Hysteresis-like compare mode, see **Section 25.2.3.3**
1010_B 1011_B, 1100_B, 1101_B 1110_B	Multi-Input Capture modes, see **Section 25.2.5**

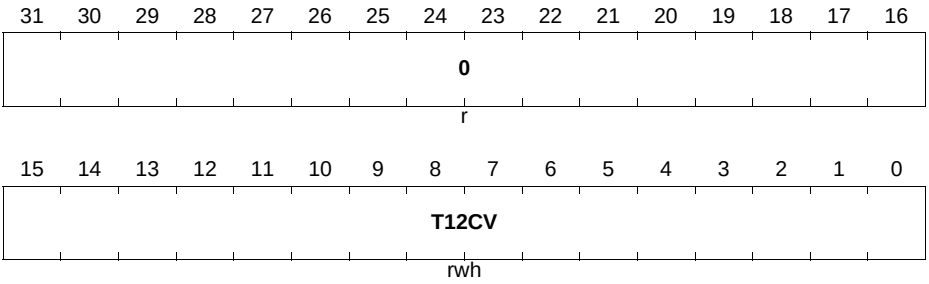
The clocking and counting scheme of the timers are controlled by the timer control registers **TCTR0** and **TCTR2**. Specific actions are triggered by write operations to register **TCTR4**.

25.2.8 T12 related Registers

25.2.8.1 T12 Counter Register

Register T12 represents the counting value of timer T12. It can only be written while the timer T12 is stopped. Write actions while T12 is running are not taken into account. Register T12 can always be read by SW.
In edge-aligned mode, T12 only counts up, whereas in center-aligned mode, T12 can count up and down.

T12
Timer T12 Counter Register **(20_H)** **Reset Value: 0000 0000_H**



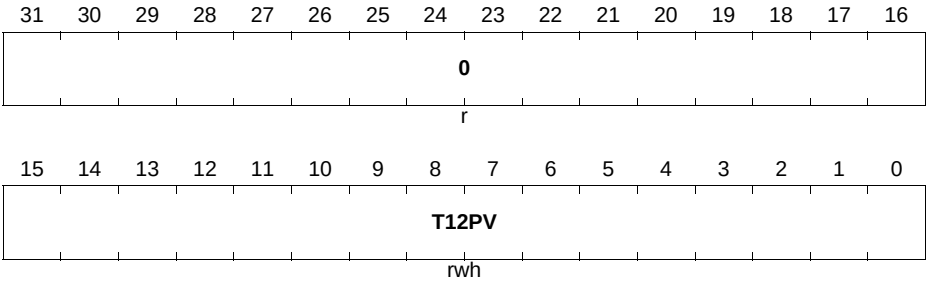
Field	Bits	Type	Description
T12CV	[15:0]	rwh	**Timer 12 Counter Value** This register represents the 16-bit counter value of Timer12.
0	[31:16]	r	**Reserved;** Returns 0 if read; should be written with 0.

Note: While timer T12 is stopped, the internal clock divider is reset in order to ensure reproducible timings and delays.

25.2.8.2 Period Register

Register T12PR contains the period value for timer T12. The period value is compared to the actual counter value of T12 and the resulting counter actions depend on the defined counting rules. This register has a shadow register and the shadow transfer is controlled by bit STE12. A read action by SW delivers the value that is currently used for the compare action, whereas the write action targets a shadow register. The shadow register structure allows a concurrent update of all T12-related values.

T12PR
Timer 12 Period Register **(24_H)** **Reset Value: 0000 0000_H**



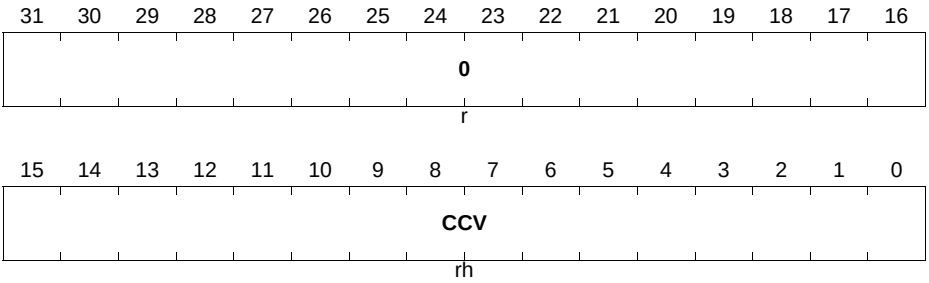
Field	Bits	Type	Description
T12PV	[15:0]	rwh	**T12 Period Value** The value T12PV defines the counter value for T12 leading to a period-match. When reaching this value, the timerT12 is set to zero (edge-aligned mode) or changes its count direction to down counting (center-aligned mode).
0	[31:16]	r	**Reserved;** Returns 0 if read; should be written with 0.

25.2.8.3 Capture/Compare Registers

In compare mode, the registers CC6xR (x = 0 - 2) are the actual compare registers for T12. The values stored in CC6xR are compared (all three channels in parallel) to the counter value of T12. In capture mode, the current value of the T12 counter register is captured by registers CC6xR if the corresponding capture event is detected.

CC6xR (x = 0-2)
Capture/Compare Register for Channel CC6x

(30_H + 4*x) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
CCV	[15:0]	rh	**Capture/Compare Value** In compare mode, the bit fields CCV contain the values, that are compared to the T12 counter value. In capture mode, the captured value of T12 can be read from these registers.
0	[31:16]	r	**Reserved;** Returns 0 if read; should be written with 0.

25.2.8.4 Capture/Compare Shadow Registers

The registers CC6xR can only be read by SW, the modification of the value is done by a shadow register transfer from register CC6xSR. The corresponding shadow registers CC6xSR can be read and written by SW. In capture mode, the value of the T12 counter register can also be captured by registers CC6xSR if the selected capture event is detected (depending on the selected capture mode).

CC6xSR (x=0-2)
Capture/Compare Shadow Reg. for Channel CC6x (40_H+4*x) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCS															
rwh															

Field	Bits	Type	Description
CCS	[15:0]	rwh	**Shadow Register for Channel x Capture/Compare Value** In compare mode, the bit fields contents of CCS are transferred to the bit fields CCV for the corresponding channel during a shadow transfer. In capture mode, the captured value of T12 can be read from these registers.
0	[31:16]	r	**Reserved;** Returns 0 if read; should be written with 0.

Note: The shadow registers can also be written by SW in capture mode. In this case, the HW capture event wins over the SW write if both happen in the same cycle (the SW write is discarded).

25.2.8.5 Dead-time Control Register

Register T12DTC controls the dead-time generation for the timer T12 compare channels. Each channel can be independently enabled/disabled for dead-time generation. If enabled, the transition from passive state to active state is delayed by the value defined by bit field DTM.
The dead time counters are clocked with the same frequency as T12.
This structure allows symmetrical dead-time generation in center-aligned and in edge-aligned PWM mode. A duty cycle of 50% leads to CC6x, COUT6x switched on for: 0.5 * period - dead time.

Note: The dead-time counters are not reset by bit T12RES, but by bit DTRES.

T12DTC
Dead-Time Control Register for Timer12

(28_H) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**DTR2**	**DTR1**	**DTR0**	**0**	**DTE2**	**DTE1**	**DTE0**	**DTM**							
r	rh	rh	rh	r	rw	rw	rw	rw							

Field	Bits	Type	Description
DTM	[7:0]	rw	**Dead-Time** Bit field DTM determines the programmable delay between switching from the passive state to the active state of the selected outputs. The switching from the active state to the passive state is not delayed.
DTE2, DTE1, DTE0	10, 9, 8	rw	**Dead Time Enable Bits** Bits DTE0..DTE2 enable and disable the dead time generation for each compare channel (0, 1, 2) of timer T12. 0_B Dead-Time Counter x is disabled. The corresponding outputs switch from the passive state to the active state (according to the actual compare status) without any delay. 1_B Dead-Time Counter x is enabled. The corresponding outputs switch from the passive state to the active state (according to the compare status) with the delay programmed in bit field DTM.
DTR2, DTR1, DTR0	14, 13, 12	rh	**Dead Time Run Indication Bits** Bits DTR0..DTR2 indicate the status of the dead time generation for each compare channel (0, 1, 2) of timer T12. 0_B Dead-Time Counter x is currently in the passive state. 1_B Dead-Time Counter x is currently in the active state.
0	11, [31:15]	r	**Reserved;** Returns 0 if read; should be written with 0.

25.2.9 Capture/Compare Control Registers

25.2.9.1 Channel State Bits

The Compare State Register CMPSTAT contains status bits monitoring the current capture and compare state and control bits defining the active/passive state of the compare channels.

CMPSTAT
Compare State Register **(60_H)** **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T13IM	COUT63PS	COUT62PS	CC62PS	COUT61PS	CC61PS	COUT60PS	CC60PS	0	CC63ST	CCPOS62	CCPOS61	CCPOS60	CC62ST	CC61ST	CC60ST
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	r	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
CC60ST, CC61ST, CC62ST, CC63ST [1]	0, 1, 2, 6	rh	**Capture/Compare State Bits** Bits CC6xST monitor the state of the capture/compare channels. Bits CC6xST (x = 0, 1, 2) are related to T12, bit CC63ST is related to T13. 0_B In compare mode, the timer count is less than the compare value. In capture mode, the selected edge has not yet been detected since the bit has been cleared by SW the last time. 1_B In compare mode, the counter value is greater than or equal to the compare value. In capture mode, the selected edge has been detected.
CCPOS60, CCPOS61, CCPOS62	3, 4, 5	rh	**Sampled Hall Pattern Bits** Bits CCPOS6x (x = 0, 1, 2) are indicating the value of the input Hall pattern that has been compared to the current and expected value. The value is sampled when the event HCRDY (Hall Compare Ready) occurs. 0_B The input CCPOSx has been sampled as 0. 1_B The input CCPOSx has been sampled as 1.
CC60PS, CC61PS, CC62PS, COUT60PS, COUT61PS, COUT62PS, COUT63PS [2]	8, 10, 12, 9, 11, 13, 14	rwh	**Passive State Select for Compare Outputs** Bits CC6xPS, COUT6xPS select the state of the corresponding compare channel, that is considered to be the passive state. During the passive state, the passive level (defined in register PSLR) is driven by the output pin. Bits CC6xPS, COUT6xPS (x = 0, 1, 2) are related to T12, bit CC63PS is related to T13. 0_B The corresponding compare signal is in passive state while CC6xST is 0. 1_B The corresponding compare signal is in passive state while CC6xST is 1. In capture mode, these bits are not used.

Field	Bits	Type	Description
T13IM 3)	15	rwh	**T13 Inverted Modulation** Bit T13IM inverts the T13 signal for the modulation of the CC6x and COUT6x (x = 0, 1, 2) signals. 0_B T13 output CC63_O is equal to CC63ST. 1_B T13 output CC63_O is equal to $\overline{\text{CC63ST}}$.
0	7, [31:16]	r	**Reserved;** Returns 0 if read; should be written with 0.

1) These bits are set and cleared according to the T12, T13 switching rules

2) These bits have shadow bits and are updated in parallel to the capture/compare registers of T12, T13 respectively. A read action targets the actually used values, whereas a write action targets the shadow bits.

3) This bit has a shadow bit and is updated in parallel to the compare and period registers of T13. A read action targets the actually used values, whereas a write action targets the shadow bit.

The Compare Status Modification Register CMPMODIF provides software-control (independent set and clear conditions) for the channel state bits CC6xST. This feature enables the user to individually change the status of the output lines by software, for example when the corresponding compare timer is stopped.

CMPMODIF
Compare State Modification Register

(64_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

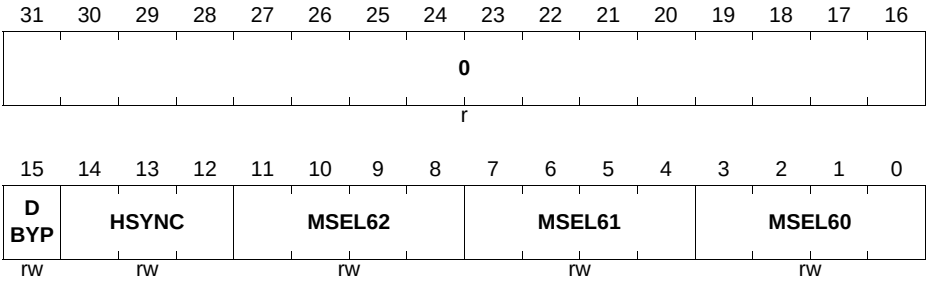
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**MCC 63R**	**0**			**MCC 62R**	**MCC 61R**	**MCC 60R**	**0**	**MCC 63S**	**0**			**MCC 62S**	**MCC 61S**	**MCC 60S**
r	w	r			w	w	w	r	w	r			w	w	w

Field	Bits	Type	Description
MCC60S, MCC61S, MCC62S, MCC63S, MCC60R, MCC61R, MCC62R, MCC63R	0, 1, 2, 6, 8, 9, 10, 14	w	**Capture/Compare Status Modification Bits** These bits are used to bits to set (MCC6xS) or to clear (MCC6xR) the corresponding bits CC6xST by SW. This feature allows the user to individually change the status of the output lines by SW, e.g. when the corresponding compare timer is stopped. This allows a bit manipulation of CC6xST-bits by a single data write action. The following functionality of a write access to bits concerning the same capture/compare state bit is provided: [MCC6xR, MCC6xS] = 00_B Bit CC6xST is not changed. 01_B Bit CC6xST is set. 10_B Bit CC6xST is cleared. 11_B reserved
0	[5:3], 7, [13:11], [31:15]	r	**Reserved;** Returns 0 if read; should be written with 0.

25.2.9.2 T12 Mode Control Register

Register T12MSEL contains control bits to select the capture/compare functionality of the three channels of Timer T12.

T12MSEL
T12 Mode Select Register **(68_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
MSEL60, MSEL61, MSEL62	[3:0], [7:4], [11:8]	rw	**Capture/Compare Mode Selection** These bit fields select the operating mode of the three T12 capture/compare channels. Each channel (x = 0, 1, 2) can be programmed individually for one of these modes (except for Hall Sensor Mode). Coding see **Table 25-5**.
HSYNC	[14:12]	rw	**Hall Synchronization** Bit field HSYNC defines the source for the sampling of the Hall input pattern and the comparison to the current and the expected Hall pattern bit fields. Coding see **Table 25-11**.
DBYP	15	rw	**Delay Bypass** DBYP controls whether the source signal for the sampling of the Hall input pattern (selected by HSYNC) is delayed by the Dead-Time Counter 0. 0_B The bypass is not active. Dead-Time Counter 0 is generating a delay after the source signal becomes active. 1_B The bypass is active. Dead-Time Counter 0 is not used for a delay.
0	[31:16]	r	**Reserved;** Returns 0 if read; should be written with 0.

25.2.9.3 Timer Control Registers

Register TCTR0 controls the basic functionality of both timers, T12 and T13.

Note: A write action to the bit fields T12CLK or T12PRE is only taken into account while the timer T12 is not running (T12R=0). A write action to the bit fields T13CLK or T13PRE is only taken into account while the timer T13 is not running (T13R=0).

TCTR0
Timer Control Register 0 **(70_H)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		STE 13	T13R	T13 PRE	T13CLK			CTM	CDIR	STE 12	T12R	T12 PRE	T12CLK		
r		rh	rh	rw	rw			rw	rh	rh	rh	rw	rw		

Field	Bits	Type	Description
T12CLK	[2:0]	rw	**Timer T12 Input Clock Select** Selects the input clock for timer T12 that is derived from the peripheral clock according to the equation $f_{T12} = f_{CC6} / 2^{<T12CLK>}$. 000_B $f_{T12} = f_{CC6}$ 001_B $f_{T12} = f_{CC6} / 2$ 010_B $f_{T12} = f_{CC6} / 4$ 011_B $f_{T12} = f_{CC6} / 8$ 100_B $f_{T12} = f_{CC6} / 16$ 101_B $f_{T12} = f_{CC6} / 32$ 110_B $f_{T12} = f_{CC6} / 64$ 111_B $f_{T12} = f_{CC6} / 128$
T12PRE	3	rw	**Timer T12 Prescaler Bit** In order to support higher clock frequencies, an additional prescaler factor of 1/256 can be enabled for the prescaler for T12. 0_B The additional prescaler for T12 is disabled. 1_B The additional prescaler for T12 is enabled.

Field	Bits	Type	Description
T12R	4	rh	**Timer T12 Run Bit[1)]** T12R starts and stops timer T12. It is set/cleared by SW by setting bits T12RR or T12RS or it is cleared by HW according to the function defined by bit field T12SSC. 0_B Timer T12 is stopped. 1_B Timer T12 is running.
STE12	5	rh	**Timer T12 Shadow Transfer Enable** Bit STE12 enables or disables the shadow transfer of the T12 period value, the compare values and passive state select bits and levels from their shadow registers to the actual registers if a T12 shadow transfer event is detected. Bit STE12 is cleared by hardware after the shadow transfer. A T12 shadow transfer event is a period-match while counting up or a one-match while counting down. 0_B The shadow register transfer is disabled. 1_B The shadow register transfer is enabled.
CDIR	6	rh	**Count Direction of Timer T12** This bit is set/cleared according to the counting rules of T12. 0_B T12 counts up. 1_B T12 counts down.
CTM	7	rw	**T12 Operating Mode** 0_B Edge-aligned Mode: T12 always counts up and continues counting from zero after reaching the period value. 1_B Center-aligned Mode: T12 counts down after detecting a period-match and counts up after detecting a one-match.

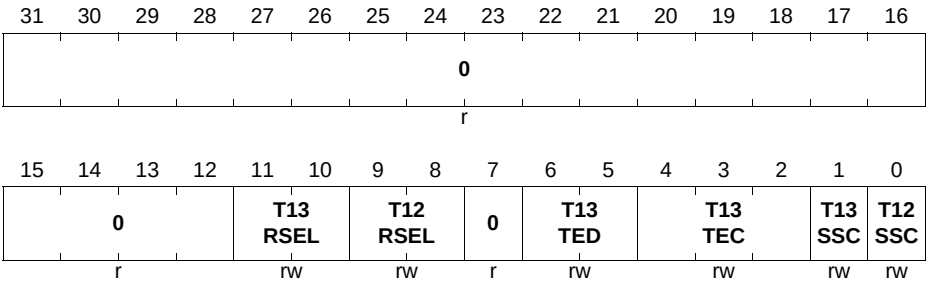
Field	Bits	Type	Description
T13CLK	[10:8]	rw	**Timer T13 Input Clock Select** Selects the input clock for timer T13 that is derived from the peripheral clock according to the equation $f_{T13} = f_{CC6} / 2^{<T13CLK>}$. 000_B $f_{T13} = f_{CC6}$ 001_B $f_{T13} = f_{CC6} / 2$ 010_B $f_{T13} = f_{CC6} / 4$ 011_B $f_{T13} = f_{CC6} / 8$ 100_B $f_{T13} = f_{CC6} / 16$ 101_B $f_{T13} = f_{CC6} / 32$ 110_B $f_{T13} = f_{CC6} / 64$ 111_B $f_{T13} = f_{CC6} / 128$
T13PRE	11	rw	**Timer T13 Prescaler Bit** In order to support higher clock frequencies, an additional prescaler factor of 1/256 can be enabled for the prescaler for T13. 0_B The additional prescaler for T13 is disabled. 1_B The additional prescaler for T13 is enabled.
T13R	12	rh	**Timer T13 Run Bit**[2)] T13R starts and stops timer T13. It is set/cleared by SW by setting bits T13RR orT13RS or it is set/cleared by HW according to the function defined by bit fields T13SSC, T13TEC and T13TED. 0_B Timer T13 is stopped. 1_B Timer T13 is running.
STE13	13	rh	**Timer T13 Shadow Transfer Enable** Bit STE13 enables or disables the shadow transfer of the T13 period value, the compare value and passive state select bit and level from their shadow registers to the actual registers if a T13 shadow transfer event is detected. Bit STE13 is cleared by hardware after the shadow transfer. A T13 shadow transfer event is a period-match. 0_B The shadow register transfer is disabled. 1_B The shadow register transfer is enabled.
0	[31:14]	r	**Reserved;** Returns 0 if read; should be written with 0.

1) A concurrent set/clear action on T12R (from T12SSC, T12RR or T12RS) will have no effect. The bit T12R will remain unchanged.

2) A concurrent set/cleared action on T13R (from T13SSC, T13TEC, T13RR or T13RS) will have no effect. The bit T12R will remain unchanged.

Register TCTR2 controls the single-shot and the synchronization functionality of both timers T12 and T13. Both timers can run in single-shot mode. In this mode they stop their counting sequence automatically after one counting period with a count value of zero. The single-shot mode and the synchronization feature of T13 to T12 allow the generation of events with a programmable delay after well-defined PWM actions of T12.

TCTR2
Timer Control Register 2 **(74_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
T12SSC	0	rw	**Timer T12 Single Shot Control** This bit controls the single shot-mode of T12. 0_B The single-shot mode is disabled, no HW action on T12R. 1_B The single shot mode is enabled, the bit T12R is cleared by HW if - T12 reaches its period value in edge-aligned mode - T12 reaches the value 1 while down counting in center-aligned mode. In parallel to the clear action of bit T12R, the bits CC6xST (x=0, 1, 2) are cleared.
T13SSC	1	rw	**Timer T13 Single Shot Control** This bit controls the single shot-mode of T13. 0_B No HW action on T13R 1_B The single-shot mode is enabled, the bit T13R is cleared by HW if T13 reaches its period value. In parallel to the clear action of bit T13R, the bit CC63ST is cleared.

Field	Bits	Type	Description
T13TEC	[4:2]	rw	**T13 Trigger Event Control** Bit field T13TEC selects the trigger event to start T13 (automatic set of T13R for synchronization to T12 compare signals) according to following combinations: 000_B no action 001_B set T13R on a T12 compare event on channel 0 010_B set T13R on a T12 compare event on channel 1 011_B set T13R on a T12 compare event on channel 2 100_B set T13R on any T12 compare event (ch. 0, 1, 2) 101_B set T13R upon a period-match of T12 110_B set T13R upon a zero-match of T12 (while counting up) 111_B set T13R on any edge of inputs CCPOSx
T13TED	[6:5]	rw	**Timer T13 Trigger Event Direction**[1)] Bit field T13TED delivers additional information to control the automatic set of bit T13R in the case that the trigger action defined by T13TEC is detected. 00_B reserved, no action 01_B while T12 is counting up 10_B while T12 is counting down 11_B independent on the count direction of T12
T12RSEL	[9:8]	rw	**Timer T12 External Run Selection** Bit field T12RSEL defines the event of signal T12HR that can set the run bit T12R by HW. 00_B The external setting of T12R is disabled. 01_B Bit T12R is set if a rising edge of signal T12HR is detected. 10_B Bit T12R is set if a falling edge of signal T12HR is detected. 11_B Bit T12R is set if an edge of signal T12HR is detected.

Field	Bits	Type	Description
T13RSEL	[11:10]	rw	**Timer T13 External Run Selection** Bit field T13RSEL defines the event of signal T13HR that can set the run bit T13R by HW. 00_B The external setting of T13R is disabled. 01_B Bit T13R is set if a rising edge of signal T13HR is detected. 10_B Bit T13R is set if a falling edge of signal T13HR is detected. 11_B Bit T13R is set if an edge of signal T13HR is detected.
0	7, [31:12]	r	**Reserved;** Returns 0 if read; should be written with 0;

1) Example:
If the timer T13 is intended to start at any compare event on T12 (T13TEC=100) the trigger event direction can be programmed to
- counting up >> a T12 channel 0, 1, 2 compare match triggers T13R only while T12 is counting up
- counting down >> a T12 channel 0, 1, 2 compare match triggers T13R only while T12 is counting down
- independent from bit CDIR >> each T12 channel 0, 1, 2 compare match triggers T13R

The timer count direction is taken from the value of bit CDIR. As a result, if T12 is running in edge-aligned mode (counting up only), T13 can only be started automatically if bit field T13TED=01 or 11.

Register TCTR4 provides software-control (independent set and clear conditions) for the run bits T12R and T13R. Furthermore, the timers can be reset (while running) and bits STE12 and STE13 can be controlled by software. Reading these bits always returns 0.

TCTR4
Timer Control Register 4 **(78_H)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T13 STD	T13 STR	T13 CNT	0		T13 RES	T13 RS	T13 RR	T12 STD	T12 STR	T12 CNT	0	DT RES	T12 RES	T12 RS	T12 RR
w	w	w	r		w	w	w	w	w	w	r	w	w	w	w

Field	Bits	Type	Description
T12RR	0	w	**Timer T12 Run Reset** Setting this bit clears the T12R bit. 0_B T12R is not influenced. 1_B T12R is cleared, T12 stops counting.
T12RS	1	w	**Timer T12 Run Set** Setting this bit sets the T12R bit. 0_B T12R is not influenced. 1_B T12R is set, T12 starts counting.
T12RES	2	w	**Timer T12 Reset** 0_B No effect on T12. 1_B The T12 counter register is cleared to zero. The switching of the output signals is according to the switching rules. Setting of T12RES has no impact on bit T12R.
DTRES	3	w	**Dead-Time Counter Reset** 0_B No effect on the dead-time counters. 1_B The three dead-time counter channels are cleared to zero.
T12CNT	5	w	**Timer T12 Count Event** 0_B No action 1_B If enabled (PISEL2), timer T12 counts one step.

Field	Bits	Type	Description
T12STR	6	w	**Timer T12 Shadow Transfer Request** 0_B No action 1_B STE12 is set, enabling the shadow transfer.
T12STD	7	w	**Timer T12 Shadow Transfer Disable** 0_B No action 1_B STE12 is cleared without triggering the shadow transfer.
T13RR	8	w	**Timer T13 Run Reset** Setting this bit clears the T13R bit. 0_B T13R is not influenced. 1_B T13R is cleared, T13 stops counting.
T13RS	9	w	**Timer T13 Run Set** Setting this bit sets the T13R bit. 0_B T13R is not influenced. 1_B T13R is set, T13 starts counting.
T13RES	10	w	**Timer T13 Reset** 0_B No effect on T13. 1_B The T13 counter register is cleared to zero. The switching of the output signals is according to the switching rules. Setting of T13RES has no impact on bit T13R.
T13CNT	13	w	**Timer T13 Count Event** 0_B No action 1_B If enabled (PISEL2), timer T13 counts one step.
T13STR	14	w	**Timer T13 Shadow Transfer Request** 0_B No action 1_B STE13 is set, enabling the shadow transfer.
T13STD	15	w	**Timer T13 Shadow Transfer Disable** 0_B No action 1_B STE13 is cleared without triggering the shadow transfer.
0	4, [12:11], [31:16]	r	**reserved;** returns 0 if read; should be written with 0;

Note: A simultaneous write of a 1 to bits that set and clear the same bit will trigger no action. The corresponding bit will remain unchanged.

25.3 Operating Timer T13

Timer T13 is implemented similarly to Timer T12, but only with one channel in compare mode. A 16-bit up-counter is connected to a channel register via a comparator, that generates a signal when the counter contents match the contents of the channel register. A variety of control functions facilitate the adaptation of the T13 structure to different application needs. In addition, T13 can be started synchronously to timer T12 events.

This section provides information about:

- T13 overview (see **Section 25.3.1**)
- Counting scheme (see **Section 25.3.2**)
- Compare mode (see **Section 25.3.3**)
- Compare output path (see **Section 25.3.4**)
- Shadow register transfer (see **Section 25.3.5**)
- T13 counter register description (see **Section 25.3.6**)

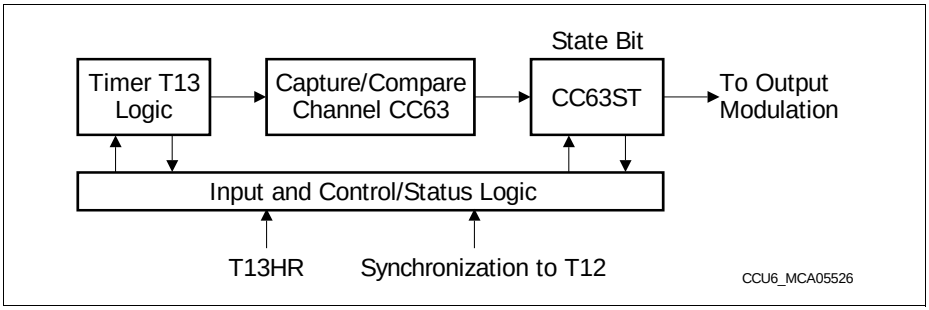


Figure 25-23 Overview Diagram of the Timer T13 Block

25.3.1 T13 Overview

Figure 25-24 shows a detailed block diagram of Timer T13. The functions of the timer T12 block are controlled by bits in registers **TCTR0**, **TCTR2**, and **PISEL2**.
Timer T13 receives its input clock, f_{T13}, from the module clock f_{CC6} via a programmable prescaler and an optional 1/256 divider or from an input signal T13HR. T13 can only count up (similar to the Edge-Aligned mode of T12).

Via a comparator, the timer T13 Counter Register **T13** is connected to the Period Register **T13PR**. This register determines the maximum count value for T13. When T13 reaches the period value, signal T13_PM (T13 Period Match) is generated and T13 is cleared to 0000_H with the next T13 clock edge. The Period Register receives a new period value from its Shadow Period Register, T13PS, that is loaded via software. The transfer of a new period value from the shadow register into T13PR is controlled via the 'T13 Shadow Transfer' control signal, T13_ST. The generation of this signal depends on the associated control bit STE13. Providing a shadow register for the period value as

well as for other values related to the generation of the PWM signal facilitates a concurrent update by software for all relevant parameters (refer to **Table 25.3.5**).
Another signal indicates whether the counter contents are equal to 0000_H (T13_ZM).

A Single-Shot control bit, T13SSC, enables an automatic stop of the timer when the current counting period is finished (see **Figure 25-26**).

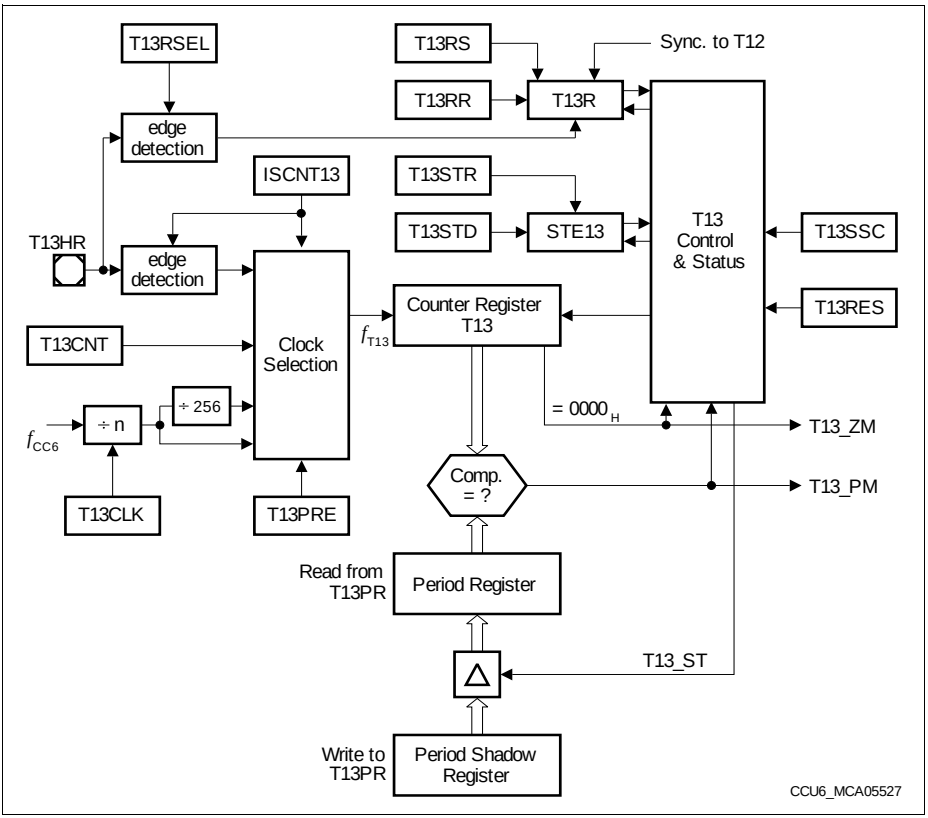


Figure 25-24 T13 Counter Logic and Period Comparators

The start or stop of T13 is controlled by the Run bit, T13R. This control bit can be set by software via the associated set/clear bits T13RS or T13RR in register **TCTR4**, or it is cleared by hardware according to preselected conditions (single-shot mode).
The timer T13 run bit T13R must not be set while the applied T13 period value is zero.
Bit T13R can be set automatically if an event of T12 is detected to synchronize T13 timings to T12 events, e.g. to generate a programmable delay via T13 after an edge of a T12 compare channel before triggering an AD conversion (T13 can trigger ADC

conversions).
Timer T13 can be cleared to 0000_H via control bit T13RES. Setting this write-only bit only clears the timer contents, but has no further effects, e.g., it does not stop the timer.

The generation of the T13 shadow transfer control signal, T13_ST, is enabled via bit STE13. This bit can be set or cleared by software indirectly through its associated set/reset control bits T13STR and T13STD.

Two bit fields, T13TEC and T13TED, control the synchronization of T13 to Timer T12 events. T13TEC selects the trigger event, while T13TED determines for which T12 count direction the trigger should be active.
While Timer T13 is running, write accesses to the count register T13 are not taken into account. If T13 is stopped, write actions to register T13 are immediately taken into account.

Note: The T13 Period Register and its associated shadow register are located at the same physical address. A write access to this address targets the Shadow Register, while a read access reads from the actual period register.

25.3.2 T13 Counting Scheme

This section describes the clocking and the counting capabilities of T13.

25.3.2.1 Clock Selection

In **Timer Mode** (**PISEL2**. ISCNT13 = 00_B), the input clock f_{T13} of Timer T13 is derived from the internal module clock f_{CC6} through a programmable prescaler and an optional 1/256 divider. The resulting prescaler factors are listed in **Table 25-6**. The prescaler of T13 is cleared while T13 is not running (**TCTR0**.T13R = 0) to ensure reproducible timings and delays.

Table 25-6 Timer T13 Input Clock Options

T13CLK	Resulting Input Clock f_{T13} Prescaler Off (T13PRE = 0)	Resulting Input Clock f_{T13} Prescaler On (T13PRE = 1)
000_B	f_{CC6}	f_{CC6} / 256
001_B	f_{CC6} / 2	f_{CC6} / 512
010_B	f_{CC6} / 4	f_{CC6} / 1024
011_B	f_{CC6} / 8	f_{CC6} / 2048
100_B	f_{CC6} / 16	f_{CC6} / 4096
101_B	f_{CC6} / 32	f_{CC6} / 8192
110_B	f_{CC6} / 64	f_{CC6} / 16384
111_B	f_{CC6} / 128	f_{CC6} / 32768

In **Counter Mode**, timer T13 counts one step:

- If a 1 is written to **TCTR4**.T13CNT and **PISEL2**.ISCNT13 = 01_B
- If a rising edge of input signal T13HR is detected and **PISEL2**.ISCNT13 = 10_B
- If a falling edge of input signal T13HR is detected and **PISEL2**.ISCNT13 = 11_B

25.3.2.2 T13 Counting

The period of the timer is determined by the value in the period Register T13PR according to the following formula:

$T13_{PER}$ = <Period-Value> + 1; in T13 clocks (f_{T13}) (25.3)

Timer T13 can only count up, comparable to the Edge-Aligned mode of T12. This leads to very simple ‘counting rule’ for the T13 counter:

- The counter is cleared with the next T13 clock edge if a Period-Match is detected. The counting direction is always upwards.

The behavior of T13 is illustrated in **Figure 25-25**.

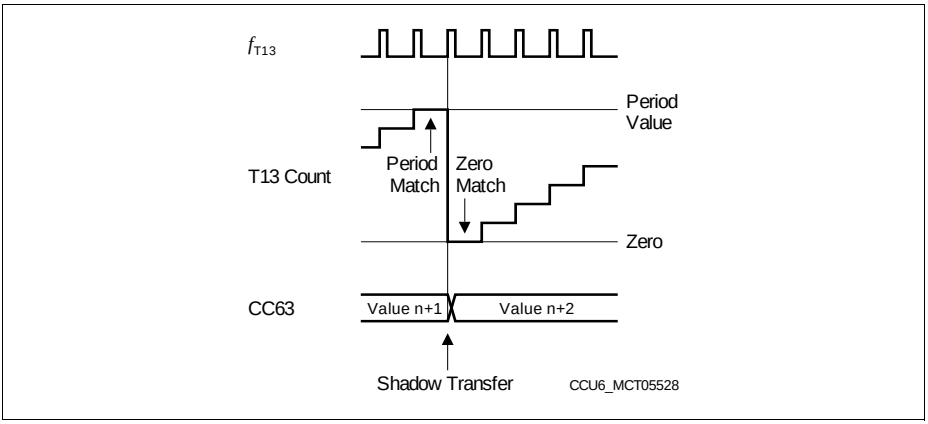


Figure 25-25 T13 Counting Sequence

25.3.2.3 Single-Shot Mode

In Single-Shot Mode, the timer run bit T13R is cleared by hardware. If bit T13SSC = 1, the timer T13 will stop when the current timer period is finished.

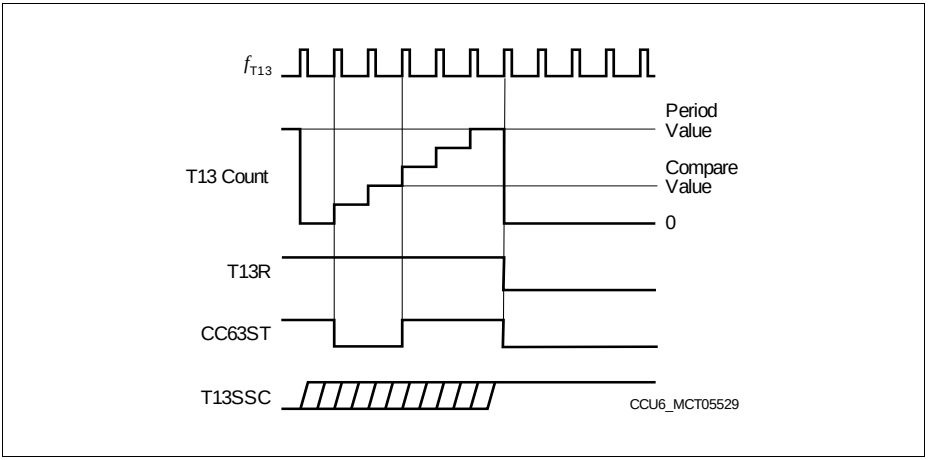


Figure 25-26 Single-Shot Operation of Timer T13

25.3.2.4 Synchronization to T12

Timer T13 can be synchronized to a T12 event. Bit fields T13TEC and T13TED select the event that is used to start Timer T13. The selected event sets bit T13R via HW, and T13 starts counting. Combined with the Single-Shot mode, this feature can be used to generate a programmable delay after a T12 event.

Figure 25-27 shows an example for the synchronization of T13 to a T12 event. Here, the selected event is a compare-match (compare value = 2) while counting up. The clocks of T12 and T13 can be different (other prescaler factor); the figure shows an example in which T13 is clocked with half the frequency of T12.

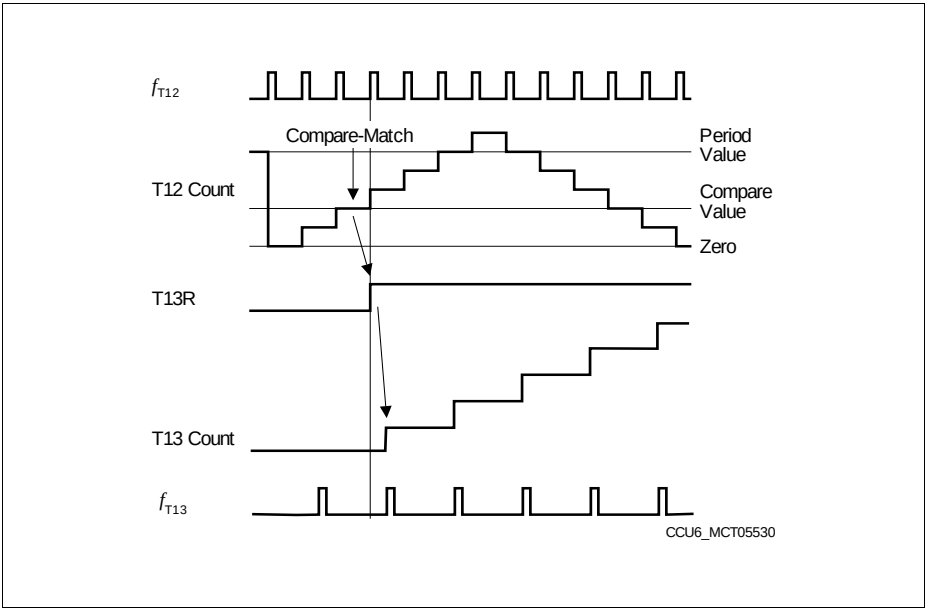


Figure 25-27 Synchronization of T13 to T12 Compare Match

Bit field T13TEC selects the trigger event to start T13 (automatic set of T13R for synchronization to T12 compare signals) according to the combinations shown in **Table 25-7**. Bit field T13TED additionally specifies for which count direction of T12 the selected trigger event should be regarded (see **Table 25-8**).

Table 25-7 T12 Trigger Event Selection

T13TEC	Selected Event
000_B	None
001_B	T12 Compare Event on Channel 0 (CM_CC60)
010_B	T12 Compare Event on Channel 1 (CM_CC61)
011_B	T12 Compare Event on Channel 2 (CM_CC62)
100_B	T12 Compare Event on any Channel (0, 1, 2)
101_B	T12 Period-Match (T12_PM)
110_B	T12 Zero-Match while counting up (T12_ZM and CDIR = 0)
111_B	Any Hall State Change

Table 25-8 T12 Trigger Event Additional Specifier

T13TED	Selected Event Specifier
00_B	Reserved, no action
01_B	Selected event is active while T12 is counting up (CDIR = 0)
10_B	Selected event is active while T12 is counting down (CDIR = 1)
11_B	Selected event is active independently of the count direction of T12

25.3.3 T13 Compare Mode

Associated with Timer T13 is one compare channel, that can perform compare operations with regard to the contents of the T13 counter.

Figure 25-23 gives an overview on the T13 channel in Compare Mode. The channel is connected to the T13 counter register via an equal-to comparator, generating a compare match signal when the contents of the counter matches the contents of the compare register.

The channel consists of the comparator and a double register structure - the actual compare register, **CC63R**, feeding the comparator, and an associated shadow register, **CC63SR**, that is preloaded by software and transferred into the compare register when signal T13 shadow transfer, T13_ST, gets active. Providing a shadow register for the compare value as well as for other values related to the generation of the PWM signal facilitates a concurrent update by software for all relevant parameters.

Associated with the channel is a State Bit, **CMPSTAT**.CC63ST, holding the status of the compare operation. **Figure 25-28** gives an overview on the logic for the State Bit.

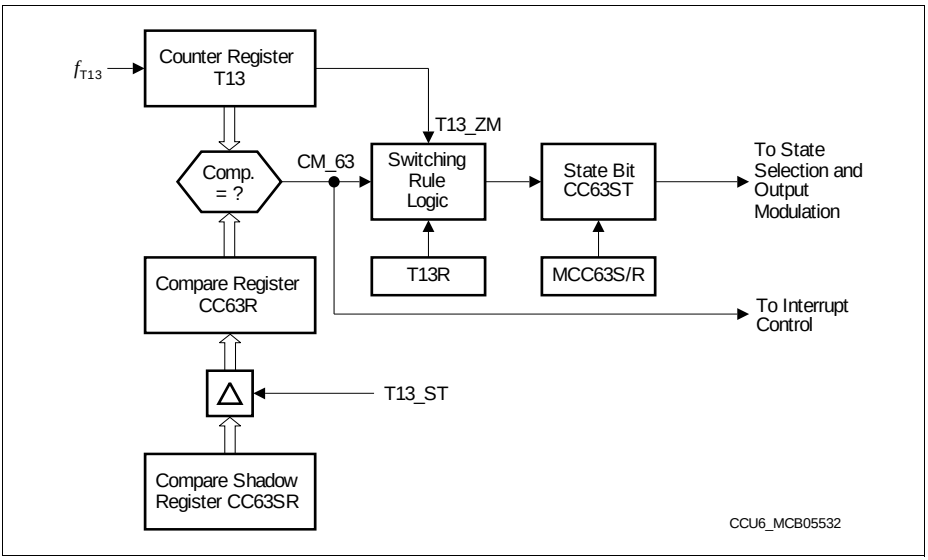


Figure 25-28 T13 State Bit Block Diagram

A compare interrupt event CM_63 is signaled when a compare match is detected. The actual setting of a State Bit has no influence on the interrupt generation.

The inputs to the switching rule logic for the CC63ST bit are the timer run bit (T13R), the timer zero-match signal (T13_ZM), and the actual individual compare-match signal CM_63. In addition, the state bit can be set or cleared by software via bits MCC63S and

MCC63R in register **CMPMODIF**.
A modification of the State Bit CC63ST by hardware is only possible while Timer T13 is running (T13R = 1). If this is the case, the following switching rules apply for setting and resetting the State Bit in Compare Mode:

State Bit **CC63ST is set** to 1

- with the next T13 clock (f_{T13}) after a compare-match (T13 is always counting up) (i.e., when the counter is incremented above the compare value);
- with the next T13 clock (f_{T13}) after a zero-match AND a parallel compare-match.

State Bit **CC63ST is cleared** to 0

- with the next T13 clock (f_{T13}) after a zero-match AND NO parallel compare-match.

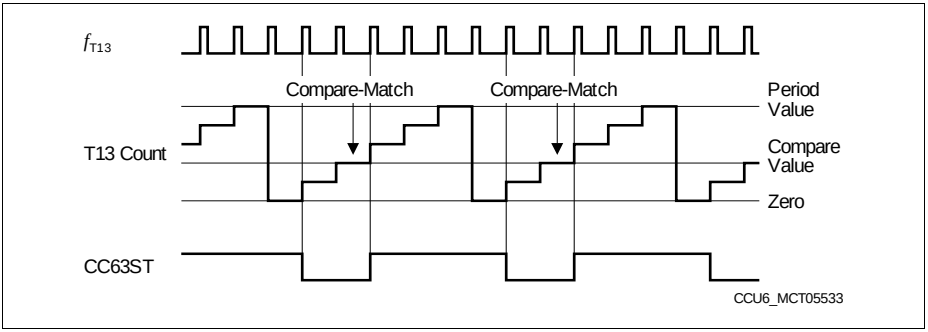


Figure 25-29 T13 Compare Operation

25.3.4 Compare Mode Output Path

Figure 25-30 gives an overview on the signal path from the channel State Bit CC63ST to its output pin COUT63. As illustrated, a user can determine the desired output behavior in relation to the current state of CC63ST. Please refer to **Section 25.2.4.3** for detailed information on the output modulation for T12 signals.

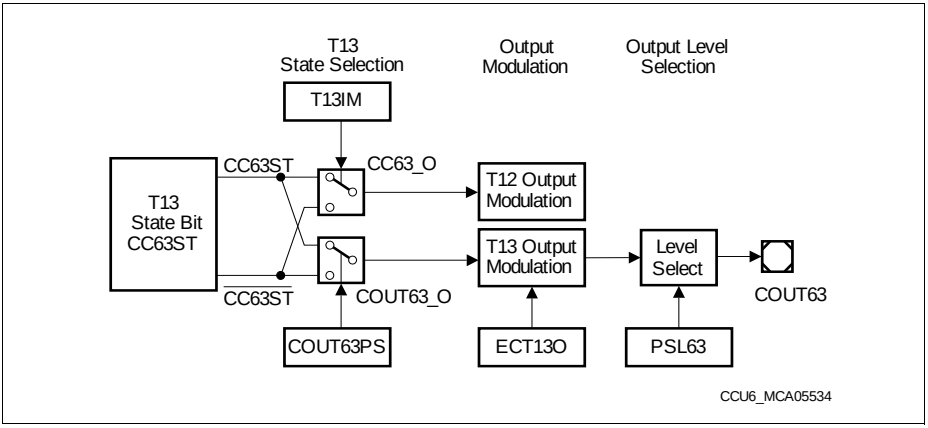


Figure 25-30 Channel 63 Output Path

The output line COUT63_O can generate a T13 PWM at the output pin COUT63. The signal CC63_O can be used to modulate the T12-related output signals with a T13 PWM. In order to decouple COUT63 from the internal modulation, the compare state leading to an active signal can be selected independently by bits T13IM and COUT63PS.

The last block of the data path is the Output Modulation block. Here, the modulation source T13 and the trap functionality are combined and control the actual level of the output pin COUT63 (see **Figure 25-31**):

- The **T13 related compare signal** COUT63_O delivered by the T13 state selection with the enable bit **MODCTR**.ECT13O
- The **trap state** TRPS with an individual enable bit **TRPCTR**.TRPEN13

If the modulation input signal COUT63_O is enabled (ECT13O = 1) and is at passive state, the modulated is also in passive state. If the modulation input is not enabled, the output is in passive state.
If the Trap State is active (TRPS = 1), then the output enabled for the trap signal (by TRPEN13 = 1) is set to the passive state.

The output of the modulation control block is connected to a level select block. It offers the option to determine the actual output level of a pin, depending on the state of the output line (decoupling of active/passive state and output polarity) as specified by the Passive State Select bit **PSLR**.PSL63. If the modulated output signal is in the passive

state, the level specified directly by PSL63 is output. If it is in the active state, the inverted level of PSL63 is output. This allows the user to adapt the polarity of an active output signal to the connected circuitry.

The PSL63 bit has a shadow register to allow for updates with the T13 shadow transfer signal (T13_ST) without undesired pulses on the output lines. A read action returns the actually used value, whereas a write action targets the shadow bit. Providing a shadow register for the PSL value as well as for other values related to the generation of the PWM signal facilitates a concurrent update by software for all relevant parameters.

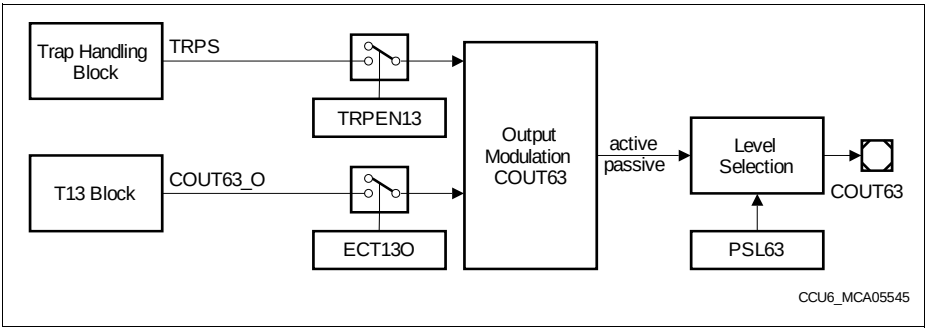


Figure 25-31 T13 Output Modulation

25.3.5 T13 Shadow Register Transfer

A special shadow transfer signal (T13_ST) can be generated to facilitate updating the period and compare values of the compare channel CC63 synchronously to the operation of T13. Providing a shadow register for values defining one PWM period facilitates a concurrent update by software for all relevant parameters. The next PWM period can run with a new set of parameters. The generation of this signal is requested by software via bit **TCTR0**.STE13 (set by writing 1 to the write-only bit **TCTR4**.T13STR, cleared by writing 1 to the write-only bit **TCTR4**.T13STD).
When signal T13_ST is active, a shadow register transfer is triggered with the next cycle of the T13 clock. Bit STE13 is automatically cleared with the shadow register transfer.
A T13 shadow register transfer takes place (T13_ST active):

- while timer T13 is not running (T13R = 0), or
- STE13 = 1 and a Period-Match is detected while T13R = 1

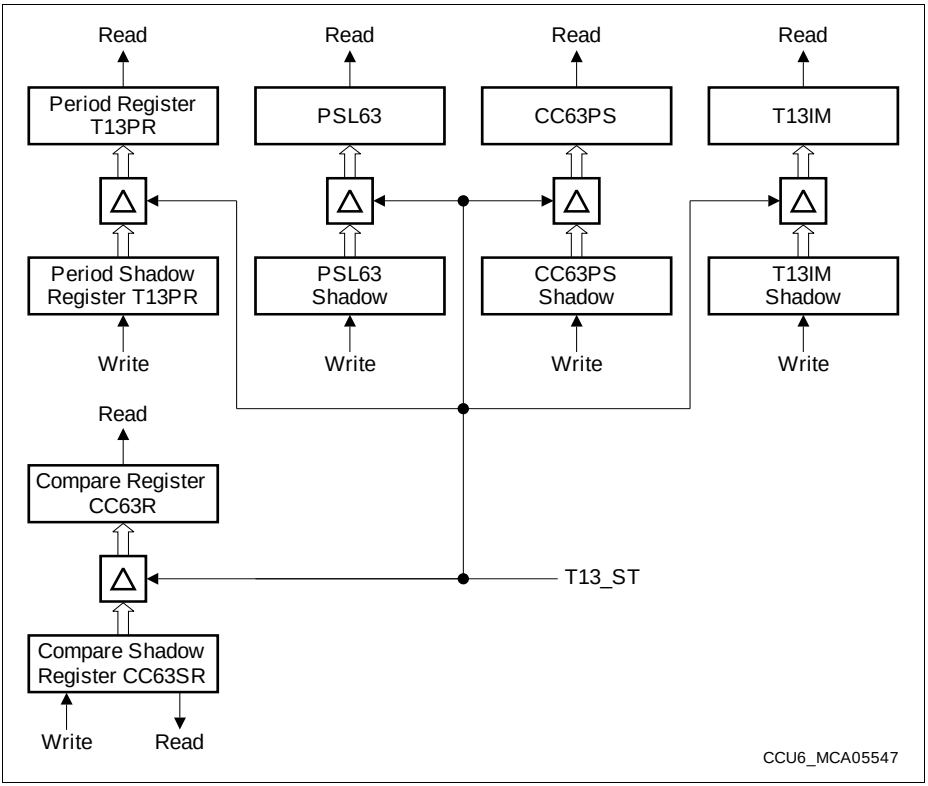


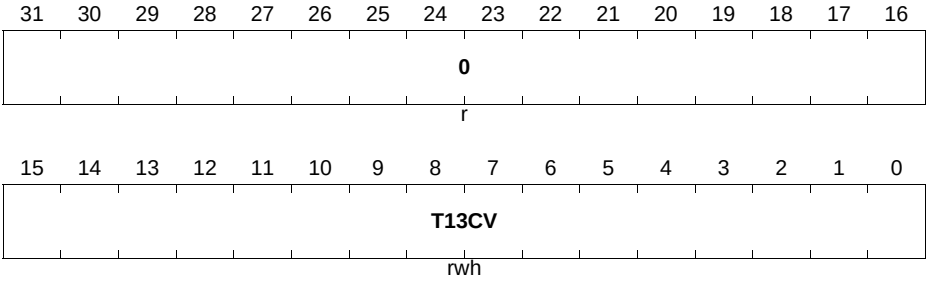
Figure 25-32 T13 Shadow Register Overview

25.3.6 T13 related Registers

25.3.6.1 T13 Counter Register

The generation of the patterns for a single channel pulse width modulation (PWM) is based on timer T13. The registers related to timer T13 can be concurrently updated (with well-defined conditions) in order to ensure consistency of the PWM signal. T13 can be synchronized to several timer T12 events.
Timer T13 only supports compare mode on its compare channel CC63.
Register T13 represents the counting value of timer T13. It can only be written while the timer T13 is stopped. Write actions while T13 is running are not taken into account. Register T13 can always be read by SW.
Timer T13 only supports edge-aligned mode (counting up).

T13
Timer T13 Counter Register **(50_H)** **Reset Value: 0000 0000_H**



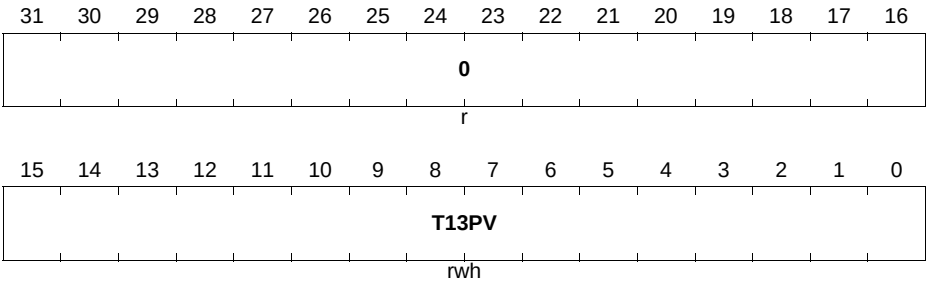
Field	Bits	Type	Description
T13CV	[15:0]	rwh	**Timer 13 Counter Value** This register represents the 16-bit counter value of Timer13.
0	[31:16]	r	**Reserved;** Returns 0 if read; should be written with 0.

Note: While timer T13 is stopped, the internal clock divider is reset in order to ensure reproducible timings and delays.

25.3.6.2 Period Register

Register T13PR contains the period value for timer T13. The period value is compared to the actual counter value of T13 and the resulting counter actions depend on the defined counting rules. This register has a shadow register and the shadow transfer is controlled by bit STE13. A read action by SW delivers the value currently used for the compare action, whereas the write action targets a shadow register. The shadow register structure allows a concurrent update of all T13-related values.

T13PR
Timer 13 Period Register **(54_H)** **Reset Value: 0000 0000_H**

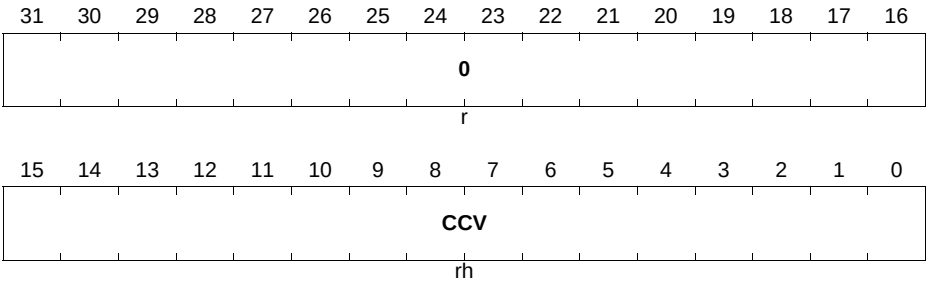


Field	Bits	Type	Description
T13PV	[15:0]	rwh	**T13 Period Value** The value T13PV defines the counter value for T13 leading to a period-match. When reaching this value, the timer T13 is set to zero.
0	[31:16]	r	**Reserved;** Returns 0 if read; should be written with 0.

25.3.6.3 Compare Register

Registers CC63R is the actual compare register for T13. The values stored in CC63R is compared to the counter value of T13. The State Bit CC63ST is located in register **CMPSTAT**.

CC63R
Compare Register for T13 **(58_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CCV	[15:0]	rh	**Channel CC63 Compare Value** The bit field CCV contains the value, that is compared to the T13 counter value.
0	[31:16]	r	**Reserved;** Returns 0 if read; should be written with 0.

25.3.6.4 Compare Shadow Register

The register CC63R can only be read by SW, the modification of the value is done by a shadow register transfer from register CC63SR. The corresponding shadow register CC63SR can be read and written by SW.

CC63SR
Compare Shadow Register for T13

($5C_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCS															
rw															

Field	Bits	Type	Description
CCS	[15:0]	rw	**Shadow Register for Channel CC63 Compare Value** The bit field contents of CCS is transferred to the bit field CCV during a shadow transfer.
0	[31:16]	r	**Reserved;** Returns 0 if read; should be written with 0.

25.4 Synchronous Start Feature

The T12 and T13 timers can be started synchronously through the T12HR and T13HR inputs of all CCU6x kernels.

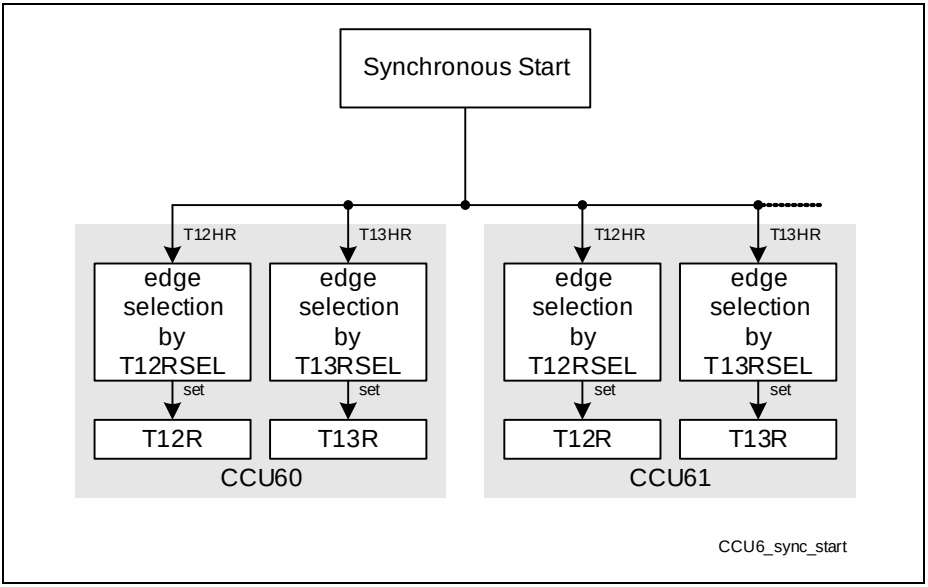


Figure 25-33 Synchronization Concept

25.5 Trap Handling

The trap functionality permits the PWM outputs to react on the state of the input signal CTRAP. This functionality can be used to switch off the power devices if the trap input becomes active (e.g. to perform an emergency stop). The trap handling and the effect on the output modulation are controlled by the bits in the trap control register **TRPCTR**. The trap flags TRPF and TRPS are located in register **IS** and can be set/cleared by SW by writing to registers **ISS** and **ISR**.
Figure 25-34 gives an overview on the trap function.

The Trap Flag TRPF monitors the trap input and initiates the entry into the Trap State. The Trap State Bit TRPS determines the effect on the outputs and controls the exit of the Trap State.
When a trap condition is detected ($\overline{\text{CTRAP}}$ = 0) and the input is enabled (TRPPEN = 1), both, the Trap Flag TRPF and the Trap State Bit TRPS, are set to 1 (trap state active). The output of the Trap State Bit TRPS leads to the Output Modulation Blocks (for T12 and for T13) and can there deactivate the outputs (set them to the passive state). Individual enable control bits for each of the six T12-related outputs and the T13-related output facilitate a flexible adaptation to the application needs.

There are a number of different ways to exit the Trap State. This offers SW the option to select the best operation for the application. Exiting the Trap State can be done either immediately when the trap condition is removed ($\overline{\text{CTRAP}}$ = 1 or TRPPEN = 0), or under software control, or synchronously to the PWM generated by either Timer T12 or Timer T13.

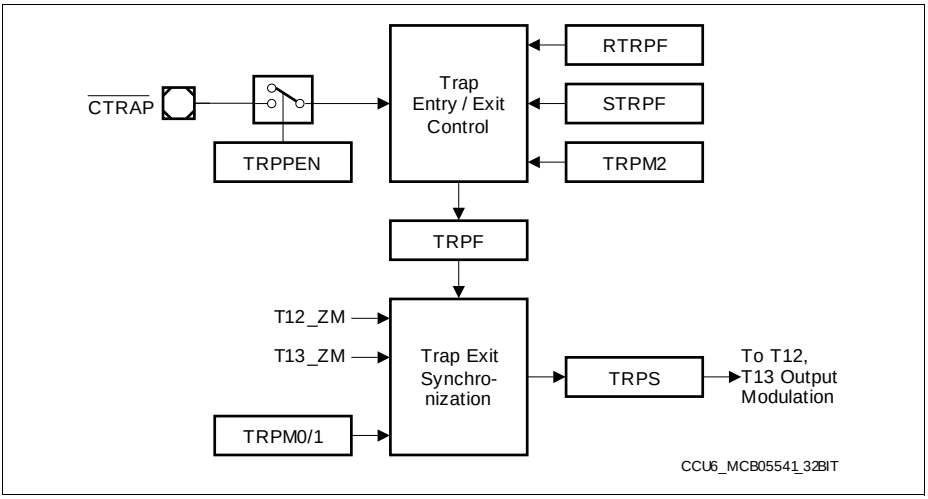


Figure 25-34 Trap Logic Block Diagram

Clearing of TRPF is controlled by the mode control bit TRPM2. If TRPM2 = 0, TRPF is automatically cleared by HW when $\overline{\text{CTRAP}}$ returns to the inactive level ($\overline{\text{CTRAP}}$ = 1) or if the trap input is disabled (TRPPEN = 0). When TRPM2 = 1, TRPF must be reset by SW after $\overline{\text{CTRAP}}$ has become inactive.

Clearing of TRPS is controlled by the mode control bits TRPM1 and TRPM0 (located in the Trap Control Register TRPCTR). A reset of TRPS terminates the Trap State and returns to normal operation. There are three options selected by TRPM1 and TRPM0. One is that the Trap State is left immediately when the Trap Flag TRPF is cleared, without any synchronization to timers T12 or T13. The other two options facilitate the synchronization of the termination of the Trap State to the count periods of either Timer T12 or Timer T13. **Figure 25-35** gives an overview on the associated operation.

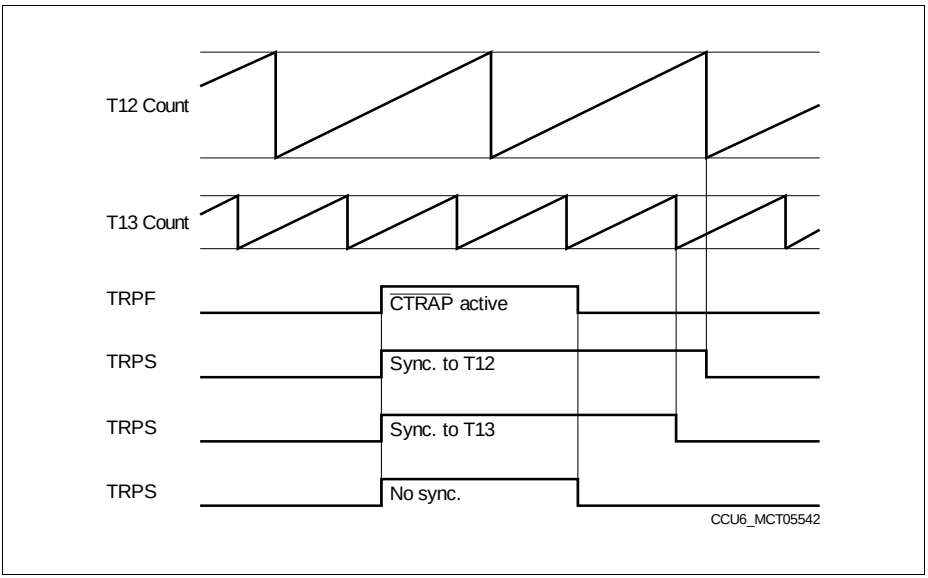


Figure 25-35 Trap State Synchronization (with TRPM2 = 0)

25.6 Multi-Channel Mode

The Multi-Channel mode offers the possibility to modulate all six T12-related output signals with one instruction. The bits in bit field **MCMOUT**.MCMP are used to specify the outputs that may become active. If Multi-Channel mode is enabled (bit **MODCTR**.MCMEN = 1), only those outputs may become active, that have a 1 at the corresponding bit position in bit field MCMP.
This bit field has its own shadow bit field **MCMOUTS**.MCMPS, that can be written by software. The transfer of the new value in MCMPS to the bit field MCMP can be triggered by, and synchronized to, T12 or T13 events. This structure permits the software to write the new value, that is then taken into account by the hardware at a well-defined moment and synchronized to a PWM signal. This avoids unintended pulses due to unsynchronized modulation sources.

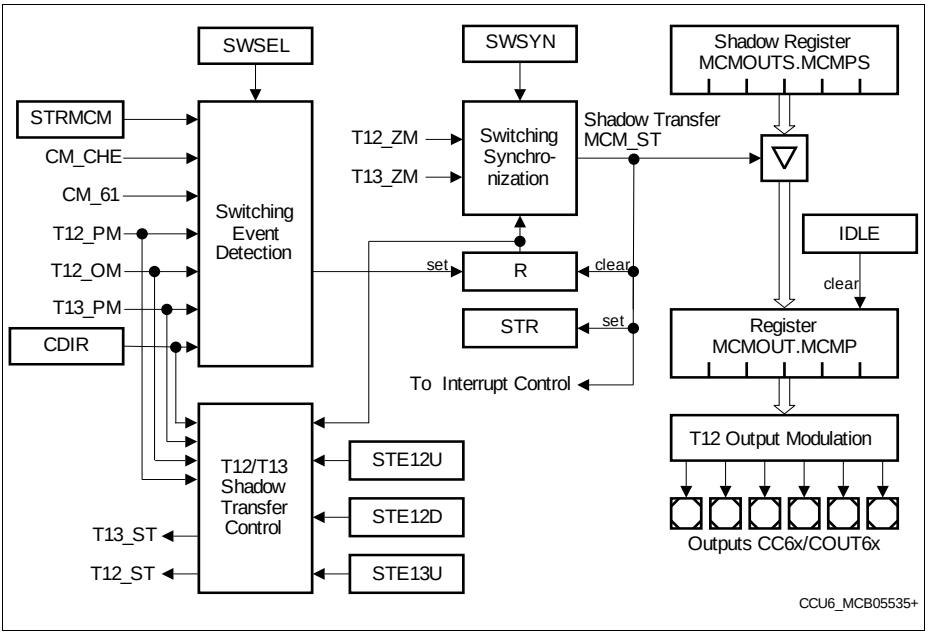


Figure 25-36 Multi-Channel Mode Block Diagram

Figure 25-36 shows the functional blocks for the Multi-Channel operation, controlled by bit fields in register **MCMCTR**. The event that triggers the update of bit field MCMP is chosen by SWSEL. In order to synchronize the update of MCMP to a PWM generated by T12 or T13, bit field SWSYN allows the selection of the synchronization event leading to the transfer from MCMPS to MCMP. Due to this structure, an update takes place with a new PWM period. A reminder flag R is set when the selected switching event occurs

(the event is not necessarily synchronous to the modulating PWM), and is cleared when the transfer takes place. This flag can be monitored by software to check for the status of this logic block. If the shadow transfer from MCMPS to MCMP takes place, bit **IS**.STR becomes set and an interrupt can be generated.
In addition to the Multi-Channel shadow transfer event MCM_ST, the shadow transfers for T12 (T12_ST) and T13 (T13_ST) can be generated to allow concurrent updates of applied duty cycles for T12 and/or T13 modulation and Multi-Channel patterns.

If it is explicitly desired, the update takes place immediately with the occurrence of the selected event when the direct synchronization mode is selected. The update can also be requested by software by writing to bit field MCMPS with the shadow transfer request bit STRMCM = 1. The option to trigger an update by SW is possible for all settings of SWSEL.
By using the direct mode and bit STRMCM = 1, the update takes place completely under software control.

Table 25-9 Multi-Channel Mode Switching Event Selection

SWSEL	Selected Event (see register MCMCTR)
000_B	No automatic event detection
001_B	Correct Hall Event (CM_CHE) detected at input signals CCPOSx without additional delay
010_B	T13 Period-Match (T13_PM)
011_B	T12 One-Match while counting down (T12_OM and CDIR = 1)
100_B	T12 Compare Channel 1 Event while counting up (CM_61 and CDIR = 0) to support the phase delay function by CC61 for block commutation mode.
101_B	T12 Period-Match while counting up (T12_PM and CDIR = 0)
110_B, 111_B	Reserved, no action

Table 25-10 Multi-Channel Mode Switching Synchronization

SWSYN	Synchronization Event (see register MCMCTR)
00_B	Direct Mode: the trigger event directly causes the shadow transfer
01_B	T13 Zero-Match (T13_ZM), the MCM shadow transfer is synchronized to a T13 PWM
10_B	T12 Zero-Match (T12_ZM), the MCM shadow transfer is synchronized to a T12 PWM
11_B	Reserved, no action

25.7 Hall Sensor Mode

For Brushless DC-Motors in block commutation mode, the Multi-Channel Mode has been introduced to provide efficient means for switching pattern generation. These patterns need to be output in relation to the angular position of the motor. For this, usually Hall sensors or Back-EMF sensing are used to determine the angular rotor position. The CCU6 provides three inputs, CCPOS0, CCPOS1, and CCPOS2, that can be used as inputs for the Hall sensors or the Back-EMF detection signals.

There is a strong correlation between the motor position and the output modulation pattern. When a certain position of the motor has been reached, indicated by the sampled Hall sensor inputs (the Hall pattern), the next, pre-determined Multi-Channel Modulation pattern has to be output. Because of different machine types, the modulation pattern for driving the motor can vary. Therefore, it is wishful to have a wide flexibility in defining the correlation between the Hall pattern and the corresponding Modulation pattern. Furthermore, a hardware mechanism significantly reduces the CPU for block-commutation.

The CCU6 offers the flexibility by having a register containing the currently assumed Hall pattern (CURH), the next expected Hall pattern (EXPH) and the corresponding output pattern (MCMP). A new Modulation pattern is output when the sampled Hall inputs match the expected ones (EXPH). To detect the next rotation phase (segment for block commutation), the CCU6 monitors the Hall inputs for changes. When the next expected Hall pattern is detected, the next corresponding Modulation pattern is output.

To increase for noise immunity (to a certain extend), the CCU6 offers the possibility to introduce a sampling delay for the Hall inputs. Some changes of the Hall inputs are not leading to the expected Hall pattern, because they are only short spikes due to noise. The Hall pattern compare logic compares the Hall inputs to the next expected pattern and also to the currently assumed pattern to filter out spikes.

For the Hall and Modulation output patterns, a double-register structure is implemented. While register **MCMOUT** holds the actually used values, its shadow register **MCMOUTS** can be loaded by software from a pre-defined table, holding the appropriate Hall and Modulation patterns for the given motor control.
A transfer from the shadow register into register MCMOUT can take place when a correct Hall pattern change is detected. Software can then load the next values into register MCMOUTS. It is also possible by software to force a transfer from MCMOUTS into MCMOUT.

Note: The Hall input signals CCPOSx and the CURH and EXPH bit fields are arranged in the following order:
CCPOS0 corresponds to CURH.0 (LSB) and EXPH.0 (LSB)
CCPOS1 corresponds to CURH.1 and EXPH.1
CCPOS2 corresponds to CURH.2 (MSB) and EXPH.2 (MSB)

25.7.1 Hall Pattern Evaluation

The Hall sensor inputs CCPOSx can be permanently monitored via an edge detection block (with the module clock f_{CC6}). In order to suppress spikes on the Hall inputs due to noise in rugged inverter environment, two optional noise filtering methods are supported by the Hall logic (both methods can be combined).

- Noise filtering with delay:
  For this function, the mode control bit fields MSEL6x for all T12 compare channels must be programmed to 1000_B and DBYP = 0. The selected event triggers Dead-Time Counter 0 to generate a programmable delay (defined by bit field DTM). When the delay has elapsed, the evaluation signal HCRDY becomes activated.
  Output modulation with T12 PWM signals is not possible in this mode.
- Noise filtering by synchronization to PWM:
  The Hall inputs are not permanently monitored by the edge detection block, but samples are taken only at defined points in time during a PWM period. This can be used to sample the Hall inputs when the switching noise (due to PWM) does not disturb the Hall input signals.

If neither the delay function of Dead-Time Counter 0 is not used for the Hall pattern evaluation nor the Hall mode for Brushless DC-Drive control is enabled, the timer T12 block is available for PWM generation and output modulation.

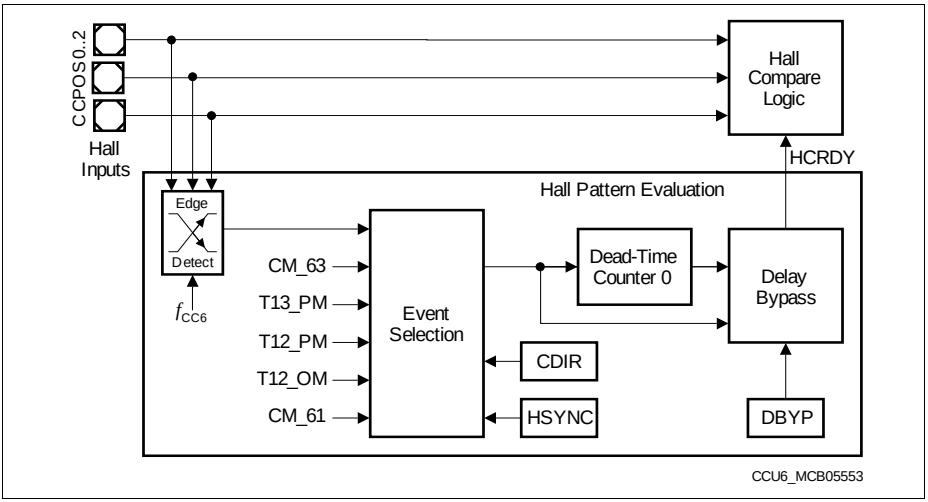


Figure 25-37 Hall Pattern Evaluation

If the evaluation signal HCRDY (Hall Compare Ready, see **Figure 25-38**) becomes activated, the Hall inputs are sampled and the Hall compare logic starts the evaluation of the Hall inputs.

Figure 25-37 illustrates the events for Hall pattern evaluation and the noise filter logic, Table 25-11 summarizes the selectable trigger input signals.

Table 25-11 Hall Sensor Mode Trigger Event Selection

HSYNC	Selected Event (see register T12MSEL)
000_B	Any edge at any of the inputs CCPOSx, independent from any PWM signal (permanent check).
001_B	A T13 Compare-Match (CM_63).
010_B	A T13 Period-Match (T13_PM).
011_B	Hall sampling triggered by HW sources is switched off.
100_B	A T12 Period-Match while counting up (T12_PM and CDIR = 0).
101_B	A T12 One-Match while counting down (T12_OM and CDIR = 1).
110_B	A T12 Compare-Match of compare channel CC61 while counting up (CM_61 and CDIR = 0).
111_B	A T12 Compare-Match of compare channel CC61 while counting down (CM_61 and CDIR = 1).

25.7.2 Hall Pattern Compare Logic

Figure 25-38 gives an overview on the double-register structure and the pattern compare logic. Software writes the next modulation pattern (MCMPS) and the corresponding current (CURHS) and expected (EXPHS) Hall patterns into the shadow register MCMOUTS. Register MCMOUT holds the actually used values CURH and EXPH. The modulation pattern MCMP is provided to the T12 Output Modulation block. The current (CURH) and expected (EXPH) Hall patterns are compared to the sampled Hall sensor inputs (visible in register **CMPSTAT**). Sampling of the inputs and the evaluation of the comparator outputs is triggered by the evaluation signal HCRDY (Hall Compare Ready), that is detailed in the next section.

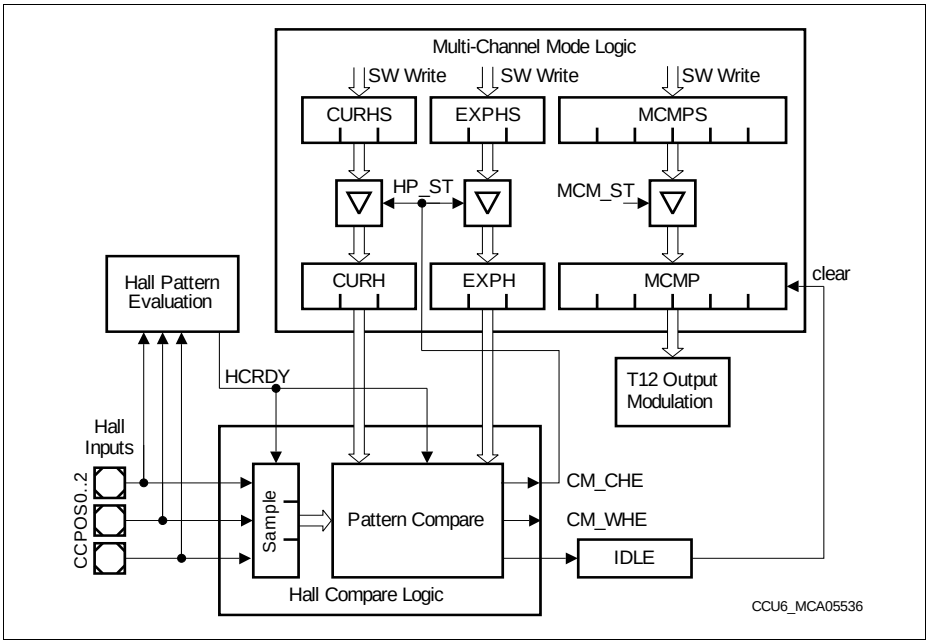


Figure 25-38 Hall Pattern Compare Logic

- If the sampled Hall pattern matches the value programmed in CURH, the detected transition was a spike (no Hall event) and no further actions are necessary.
- If the sampled Hall pattern matches the value programmed in EXPH, the detected transition was the expected event (correct Hall event CM_CHE) and the MCMP value has to change.
- If the sampled Hall pattern matches neither CURH nor EXPH, the transition was due to a major error (wrong Hall event CM_CWE) and can lead to an emergency shut down (IDLE).

At every correct Hall event (CM_CHE), the next Hall patterns are transferred from the shadow register MCMOUTS into MCMOUT (Hall pattern shadow transfer HP_ST), and a new Hall pattern with its corresponding output pattern can be loaded (e.g. from a predefined table in memory) by software into MCMOUTS. For the Modulation patterns, signal MCM_ST is used to trigger the transfer.
Loading this shadow register can also be done by writing MCMOUTS.STRHP = 1 (for EXPH and CURH) or MCMOUTS.STRMCMP = 1 (for MCMP).

25.7.3 Hall Mode Flags

Depending on the Hall pattern compare operation, a number of flags are set in order to indicate the status of the module and to trigger further actions and interrupt requests.

Flag **IS**.CHE (Correct Hall Event) is set by signal CM_CHE when the sampled Hall pattern matches the expected one (EXPH). This flag can also be set by SW by setting bit **ISS**.SCHE = 1. If enabled by bit **IEN**.ENCHE = 1, the set signal for CHE can also generate an interrupt request to the CPU. Bit field **INP**.INPCHE defines which service request output becomes activated in case of an interrupt request.To clear flag CHE, SW needs to write **ISR**.RCHE = 1.
Flag IS.WHE indicates a Wrong Hall Event. Its handling for flag setting and resetting as well as interrupt request generation are similar to the mechanism for flag CHE.
The implementation of flag STR is done in the same way as for CHE and WHE. This flag is set by HW by the shadow transfer signal MCM_ST (see also **Figure 25-36**).

Please note that for flags CHE, WHE, and STR, the interrupt request generation is triggered by the set signal for the flag. That means, a request can be generated even if the flag is already set. There is no need to clear the flag in order to enable further interrupt requests.

The implementation for the IDLE flag is different. It is set by HW through signal CM_WHE if enabled by bit ENIDLE. Software can also set the flag via bit SIDLE. As long as bit IDLE is set, the modulation pattern field MCMP is cleared to force the outputs to the passive state. Flag IDLE must be cleared by software by writing RIDLE = 1 in order to return to normal operation. To fully restart from IDLE mode, the transfer requests for the bit fields in register MCMOUTS to register MCMOUT have to be initiated by software via bits STRMCM and STRHP in register MCMOUTS. In this way, the release from IDLE mode is under software control, but can be performed synchronously to the PWM signal.

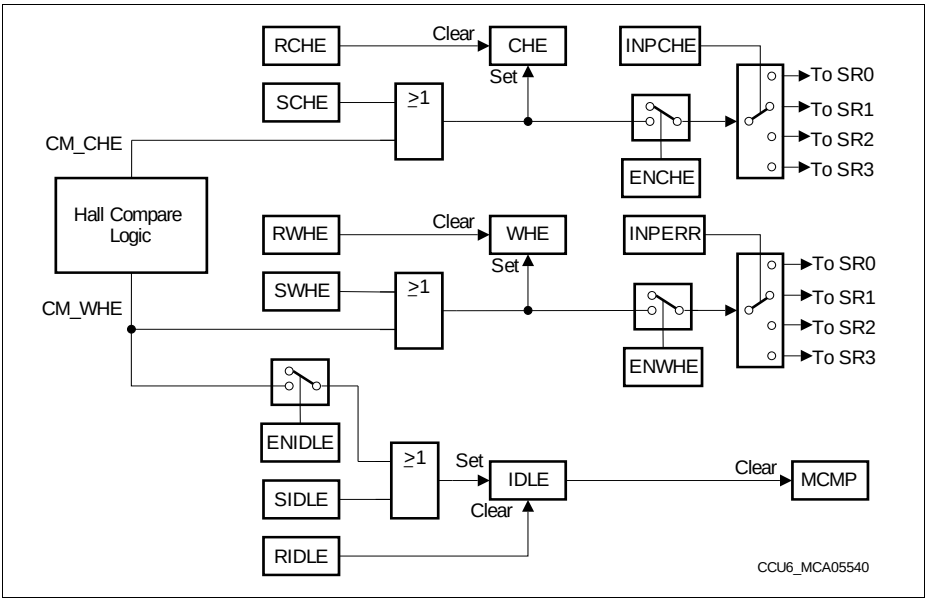


Figure 25-39 Hall Mode Flags

25.7.4 Hall Mode for Brushless DC-Motor Control

The CCU6 provides a mode for the Timer T12 Block especially targeted for convenient control of block commutation patterns for Brushless DC-Motors. This mode is selected by setting all **T12MSEL**.MSEL6x bit fields of the three T12 Channels to 1000_B.
In this mode, illustrated in **Figure 25-40**, channel CC60 is placed in capture mode to measure the time elapsed between the last two correct Hall events, channel CC61 in compare mode to provide a programmable phase delay between the Hall event and the application of a new PWM output pattern, and channel CC62 also in compare mode as first time-out criterion. A second time-out criterion can be built by the T12 period match event.

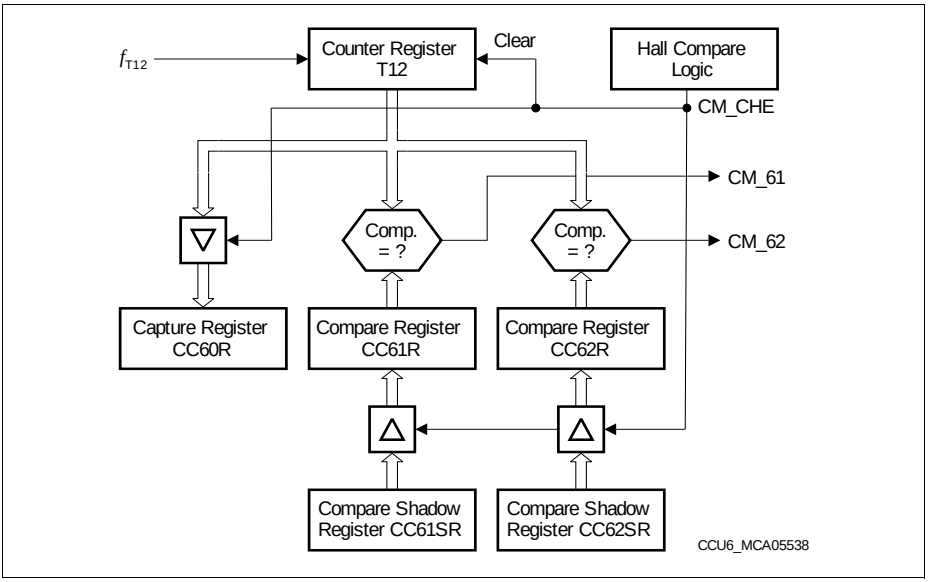


Figure 25-40 T12 Block in Hall Sensor Mode

The signal CM_CHE from the Hall compare logic is used to transfer the new compare values from the shadow registers CC6xSR into the actual compare registers CC6xR, performs the shadow transfer for the T12 period register, to capture the current T12 contents into register CC60R, and to clear T12.

Note: In this mode, the shadow transfer signal T12_ST is not generated. Not all shadow bits, such as the PSLy bits, will be transferred to their main registers. To program the main registers, SW needs to write to these registers while Timer T12 is stopped. In this case, a SW write actualizes both registers.

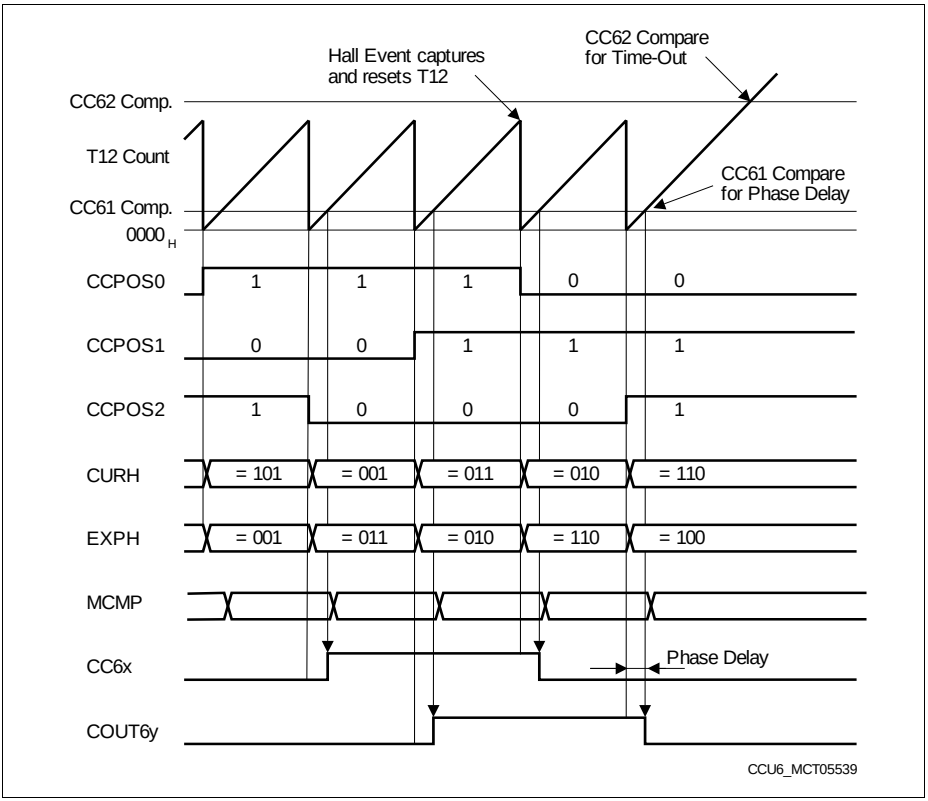


Figure 25-41 Brushless DC-Motor Control Example (all MSEL6x = 1000_B)

After the detection of an expected Hall pattern (CM_CHE active), the T12 count value is captured into channel CC60 (representing the actual rotor speed by measuring the elapsed time between the last two correct Hall events), and T12 is reset. When the timer reaches the compare value in channel CC61, the next multi-channel state is switched by triggering the shadow transfer of bit field MCMP (if enabled in bit field **SWEN**). This trigger event can be combined with the synchronization of the next multi-channel state to the PWM source (to avoid spikes on the output lines, see **Section 25.6**). This compare function of channel CC61 can be used as a phase delay from the position sensor input signals to the switching of the output signals, that is necessary if a sensorless back-EMF technique or Hall sensors are used. The compare value in channel CC62 can be used as a time-out trigger (interrupt), indicating that the actual motor speed is far below the desired destination value. An abnormal load change can be detected with this feature and PWM generation can be disabled.

25.8 Modulation Control Registers

25.8.1 Modulation Control

This register contains bits enabling the modulation of the corresponding output signal by PWM pattern generated by the timers T12 and T13. Furthermore, the multi-channel mode can be enabled as additional modulation source for the output signals.

MODCTR
Modulation Control Register **(80_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECT13O	0	T13MODEN						MCMEN	0	T12MODEN					
rw	r	rw						rw	r	rw					

Field	Bits	Type	Description
T12MODEN	[5:0]	rw	**T12 Modulation Enable** These bits enable the modulation of the corresponding output signal by a PWM pattern generated by timer T12. T12MODEN0 = MODCTR.0 for output CC60 T12MODEN1 = MODCTR.1 for output COUT60 T12MODEN2 = MODCTR.2 for output CC61 T12MODEN3 = MODCTR.3 for output COUT61 T12MODEN4 = MODCTR.4 for output CC62 T12MODEN5 = MODCTR.5 for output COUT62 0_B The modulation of the corresponding output signal by a T12 PWM pattern is disabled. 1_B The modulation of the corresponding output signal by a T12 PWM pattern is enabled.
MCMEN	7	rw	**Multi-Channel Mode Enable** 0_B The modulation of the corresponding output signal by a multi-channel pattern according to bit field MCMOUT is disabled. 1_B The modulation of the corresponding output signal by a multi-channel pattern according to bit field MCMOUT is enabled.
T13MODEN	[13:8]	rw	**T13 Modulation Enable** These bits enable the modulation of the corresponding output signal by the PWM pattern CC63_O generated by timer T13. T13MODEN0 = MODCTR.8 for output CC60 T13MODEN1 = MODCTR.9 for output COUT60 T13MODEN2 = MODCTR.10 for output CC61 T13MODEN3 = MODCTR.11 for output COUT61 T13MODEN4 = MODCTR.12 for output CC62 T13MODEN5 = MODCTR.13 for output COUT62 0_B The modulation of the corresponding output signal by a T13 PWM pattern is disabled. 1_B The modulation of the corresponding output signal by a T13 PWM pattern is enabled.

Field	Bits	Type	Description
ECT13O	15	rw	**Enable Compare Timer T13 Output** 0_B The output COUT63 is in the passive state. 1_B The output COUT63 is enabled for the PWM signal generated by T13.
0	6, 14, [31:16]	r	**Reserved;** Returns 0 if read; should be written with 0.

25.8.2 Trap Control Register

The register TRPCTR controls the trap functionality. It contains independent enable bits for each output signal and control bits to select the behavior in case of a trap condition. The trap condition is a low level on the $\overline{\text{CTRAP}}$ input pin, that is monitored (inverted level) by bit IS.TRPF. While TRPF=1 (trap input active), the trap state bit IS.TRPS is set to 1.

TRPCTR
Trap Control Register **(84_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRPPEN	TRPEN13	TRPEN						0					TRPM2	TRPM1	TRPM0
rw	rw	rw						r					rw	rw	rw

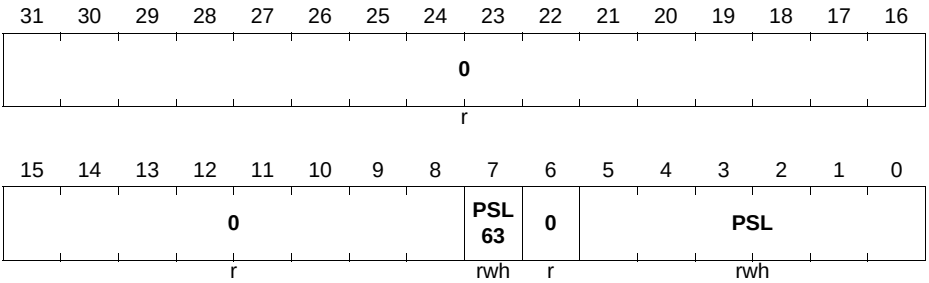
Field	Bits	Type	Description
TRPM1, TRPM0	1, 0	rw	**Trap Mode Control Bits 1, 0** These two bits define the behavior of the selected outputs when leaving the trap state after the trap condition has become inactive again. A synchronization to the timer driving the PWM pattern avoids unintended pulses when leaving the trap state. The combination [TRPM1, TRPM0] leads to: 00_B The trap state is left (return to normal operation) after TRPF has become 0 again when a zero-match of T12 (while counting up) is detected (synchronization to T12). 01_B The trap state is left (return to normal operation) after TRPF has become 0 again when a zero-match of T13 is detected (synchronization to T13). 10_B reserved 11_B The trap state is left (return to normal operation) immediately after TRPF has become 0 again without any synchronization to T12 or T13.
TRPM2	2	rw	**Trap Mode Control Bit 2** This bit defines how the trap flag TRPF can be cleared after the trap input condition ($\overline{\text{CTRAP}}$ = 0 and TRPPEN = 1) is no longer valid (either by $\overline{\text{CTRAP}}$ = 1 or by TRPPEN = 0). 0_B Automatic Mode: Bit TRPF is cleared by HW if the trap input condition is no longer valid. 1_B Manual Mode: Bit TRPF stays 0 after the trap input condition is no longer valid. It has to be cleared by SW by writing ISR.RTRPF = 1.

Field	Bits	Type	Description
TRPEN	[13:8]	rw	**Trap Enable Control** Setting a bit enables the trap functionality for the following corresponding output signals: TRPEN0 = TRPCTR.8 for output CC60 TRPEN1 = TRPCTR.9 for output COUT60 TRPEN2 = TRPCTR.10 for output CC61 TRPEN3 = TRPCTR.11 for output COUT61 TRPEN4 = TRPCTR.12 for output CC62 TRPEN5 = TRPCTR.13 for output COUT62 0_B The trap functionality of the corresponding output signal is disabled. The output state is independent from bit IS.TRPS. 1_B The trap functionality of the corresponding output signal is enabled. The output state is set to the passive while IS.TRPS=1.
TRPEN13	14	rw	**Trap Enable Control for Timer T13** 0_B The trap functionality for output COUT63 is disabled. The output state is independent from bit IS.TRPS. 1_B The trap functionality for output COUT63 is enabled. The output state is set to the passive while IS.TRPS=1.
TRPPEN	15	rw	**Trap Pin Enable** This bit enables the input (pin) function for the trap generation. An interrupt can only be generated if a falling edge is detected at pin $\overline{\text{CTRAP}}$ while TRPPEN = 1. 0_B The CCU6 trap functionality based on the input $\overline{\text{CTRAP}}$ is disabled. A CCU6 trap can only be generated by SW by setting bit TRPF. 1_B The CCU6 trap functionality based on the input $\overline{\text{CTRAP}}$ is enabled. A CCU6 trap can be generated by SW by setting bit TRPF or by $\overline{\text{CTRAP}}$=0.
0	[7:3], [31:16]	r	**Reserved;** Returns 0 if read; should be written with 0.

25.8.3 Passive State Level Register

Register PSLR defines the passive state level of the PWM outputs of the module. The passive state level is the value that is driven during the passive state of the output. During the active state, the corresponding output pin drives the active state level, that is the inverted passive state level. The passive state level permits to adapt the driven output levels to the driver polarity (inverted, not inverted) of the connected power stage. The bits in this register have shadow bit fields to permit a concurrent update of all PWM-related parameters (bit field PSL is updated with T12_ST, whereas PSL63 is updated with T13_ST). The actually used values can be read (attribute “rh”), whereas the shadow bits can only be written (attribute “w”).

PSLR
Passive State Level Register **(88_H)** **Reset Value: $0000\ 0000_H$**



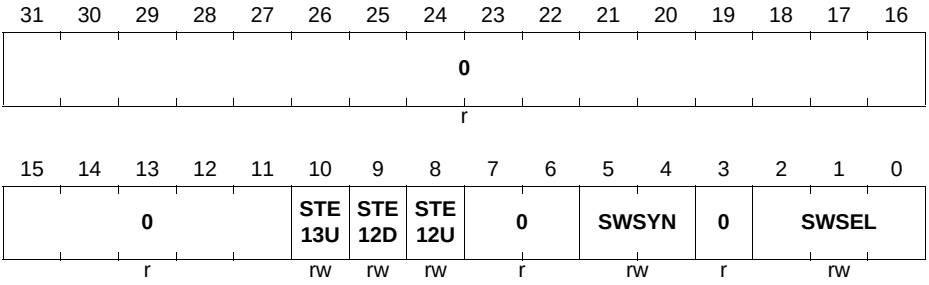
Field	Bits	Type	Description
PSL	[5:0]	rwh	**Compare Outputs Passive State Level** These bits define the passive level driven by the module outputs during the passive state. PSL0 = PSLR.0 for output CC60 PSL1 = PSLR.1 for output COUT60 PSL2 = PSLR.2 for output CC61 PSL3 = PSLR.3 for output COUT61 PSL4 = PSLR.4 for output CC62 PSL5 = PSLR.5 for output COUT62 0_B The passive level is 0. 1_B The passive level is 1.
PSL63	7	rwh	**Passive State Level of Output COUT63** This bit defines the passive level driven by the module output COUT63 during the passive state. 0_B The passive level is 0. 1_B The passive level is 1.
0	6, [31:8]	r	**Reserved;** Returns 0 if read; should be written with 0.

25.8.4 Multi-Channel Mode Registers

Register MCMCTR contains control bits for the multi-channel functionality.

MCMCTR
Multi-Channel Mode Control Register

(94_H) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
SWSEL	[2:0]	rw	**Switching Selection** Bit field SWSEL selects one of the following trigger request sources (next multi-channel event) for the shadow transfer MCM_ST from MCMPS to MCMP. The trigger request is stored in the reminder flag R until the shadow transfer is done and flag R is cleared automatically with the shadow transfer. The shadow transfer takes place synchronously with an event selected in bit field SWSYN. 000_B No trigger request will be generated 001_B Correct Hall pattern detected (CM_CHE) 010_B T13 period-match detected (while counting up) 011_B T12 one-match (while counting down) 100_B T12 channel 1 compare-match detected (phase delay function) 101_B T12 period match detected (while counting up) 110_B reserved, no trigger request will be generated 111_B reserved, no trigger request will be generated
SWSYN	[5:4]	rw	**Switching Synchronization** Bit field SWSYN defines the synchronization mechanism of the shadow transfer event MCM_ST if it has been requested before (flag R set by an event selected by SWSEL) and if MCMEN = 1. This feature permits the synchronization of the outputs to the PWM source, that is used for modulation (T12 or T13). 00_B Direct; the trigger event immediately leads to the shadow transfer 01_B A T13 zero-match triggers the shadow transfer 10_B A T12 zero-match (while counting up) triggers the shadow transfer 11_B reserved; no action
STE12U	8	rw	**Shadow Transfer Enable for T12 Upcounting** This bit enables the shadow transfer T12_ST if flag MCMOUT.R is set or becomes set while a T12 period match is detected while counting up. 0_B No action 1_B The T12_ST shadow transfer mechanism is enabled if MCMEN = 1.

Field	Bits	Type	Description
STE12D	9	rw	**Shadow Transfer Enable for T12 Downcounting** This bit enables the shadow transfer T12_ST if flag MCMOUT.R is set or becomes set while a T12 one match is detected while counting down. 0_B No action 1_B The T12_ST shadow transfer mechanism is enabled if MCMEN = 1.
STE13U	10	rw	**Shadow Transfer Enable for T13 Upcounting** This bit enables the shadow transfer T13_ST if flag MCMOUT.R is set or becomes set while a T13 period match is detected. 0_B No action 1_B The T13_ST shadow transfer mechanism is enabled if MCMEN = 1.
0	3, [7:6], [31:11]	r	**Reserved;** Returns 0 if read; should be written with 0.

Register MCMOUTS contains bits used as pattern input for the multi-channel mode and the Hall mode. This register is a shadow register (that can be read and written) for register MCMOUT, indicating the currently active signals.

MCMOUTS
Multi-Channel Mode Output Shadow Register

($8C_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

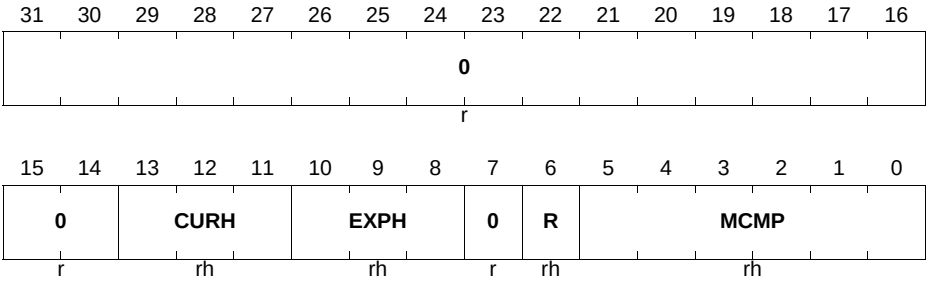
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STRHP	0	CURHS			EXPHS			STRMCM	0	MCMPS					
w	r	rw			rw			w	r	rw					

Field	Bits	Type	Description
MCMPS	[5:0]	rw	**Multi-Channel PWM Pattern Shadow** Bit field MCMPS is the shadow bit field for bit field MCMP. The multi-channel shadow transfer is triggered by MCM_ST according to the transfer conditions defined by register MCMCTR.
STRMCM	7	w	**Shadow Transfer Request for MCMPS** Writing STRMCM = 1 leads to an immediate activation of MCM_ST to update bit field MCMP by the value of MCMPS. When read, this bit always delivers 0. 0_B No action. 1_B Bit field MCMP is updated.
EXPHS	[10:8]	rw	**Expected Hall Pattern Shadow** Bit field EXPHS is the shadow bit field for bit field EXPH. The shadow transfer takes place when a correct Hall event is detected (CM_CHE).
CURHS	[13:11]	rw	**Current Hall Pattern Shadow** Bit field CURHS is the shadow bit field for bit field CURH. The shadow transfer takes place when a correct Hall event is detected (CM_CHE).

Field	Bits	Type	Description
STRHP	15	w	**Shadow Transfer Request for the Hall Pattern** Writing STRHP = 1 leads to an immediate activation of HP_ST to update bit fields EXPH and CURH by EXPHS and CURHS. When read, this bit always delivers 0. 0_B No action. 1_B Bit fields EXPH and CURH are updated.
0	6, 14, [31:16]	r	**Reserved;** Returns 0 if read; should be written with 0.

MCMOUT
Multi-Channel Mode Output Register

(90_H) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
MCMP	[5:0]	rh	**Multi-Channel PWM Pattern** Bit field MCMP defines the output pattern for the multi-channel mode. If this mode is enabled by MODCTR.MCMEN = 1, the output state of all T12 related PWM outputs can be modified. This bit field is 0 while IS.IDLE = 1. MCMP0 = MCMOUT.0 for output CC60 MCMP1 = MCMOUT.1 for output COUT60 MCMP2 = MCMOUT.2 for output CC61 MCMP3 = MCMOUT.3 for output COUT61 MCMP4 = MCMOUT.4 for output CC62 MCMP5 = MCMOUT.5 for output COUT62 0_B The output is set to the passive state. A PWM generated by T12 or T13 are not taken into account. 1_B The output can be in the active state, depending on the enabled PWM modulation signals generated by T12, T13 and the trap state.
R	6	rh	**Reminder Flag** This flag indicates that the shadow transfer from MCMPS to MCMP has been requested by the selected trigger source. It is cleared when the shadow transfer takes place or while MCMEN=0. 0_B A shadow transfer MCM_ST is not requested. 1_B A shadow transfer MCM_ST is requested, but has not yet been executed, because the selected synchronization condition has not yet occurred.
EXPH	[10:8]	rh	**Expected Hall Pattern** Bit field EXPH is updated by a shadow transfer HP_ST from bit field EXPHS. If HCRDY = 1, EXPH is compared to the sampled CCPOSx inputs in order to detect the occurrence of the next desired (=expected) hall pattern or a wrong pattern. If the sampled hall pattern at the hall input pins is equal to bit field EXPH, a correct Hall event has been detected (CM_CHE).

Field	Bits	Type	Description
CURH	[13:11]	rh	**Current Hall Pattern** Bit field CURH is updated by a shadow transfer HP_ST from bit field CURHS. If HCRDY = 1, CURH is compared to the sampled CCPOSx inputs in order to detect a spike. If the sampled Hall pattern at the Hall input pins is equal to bit field CURH, no Hall event has been detected. If the sampled Hall input pattern is neither equal to CURH nor equal to EXPH, the Hall event was not the desired one and may be due to a fatal error (e.g. blocked rotor, etc.). In this case, a wrong Hall event has been detected (CM_WHE).
0	7, [31:14]	r	**Reserved;** Returns 0 if read; should be written with 0.

25.9 Interrupt Handling

This section describes the interrupt handling of the CCU6 module.

25.9.1 Interrupt Structure

The HW interrupt event or the SW setting of the corresponding interrupt set bit (in register ISS) sets the event indication flags (in register IS) and can trigger the interrupt generation. The interrupt pulse is generated independently from the interrupt status flag in register IS (it is not necessary to clear the related status bit to be able to generate another interrupt). The interrupt flag can be cleared by SW by writing to the corresponding bit in register ISR.

If enabled by the related interrupt enable bit in register IEN, an interrupt pulse can be generated on one of the four service request outputs (SR0 to SR3) of the module. If more than one interrupt source is connected to the same interrupt node pointer (in register INP), the requests are logically OR-combined to one common service request output (see **Figure 25-42**).

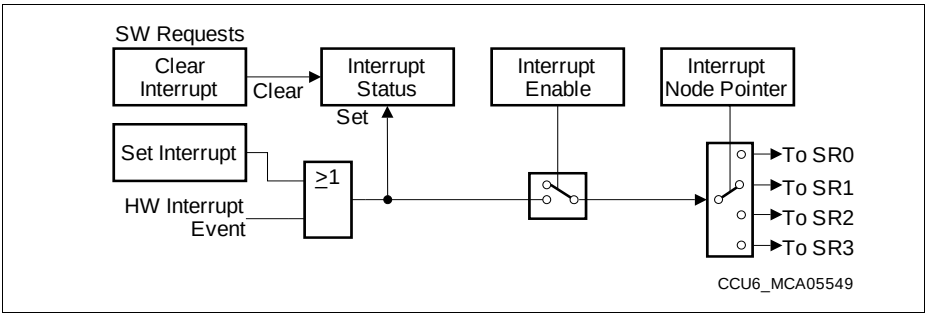


Figure 25-42 General Interrupt Structure

The available interrupt events in the CCU6 are shown in **Figure 25-43**.

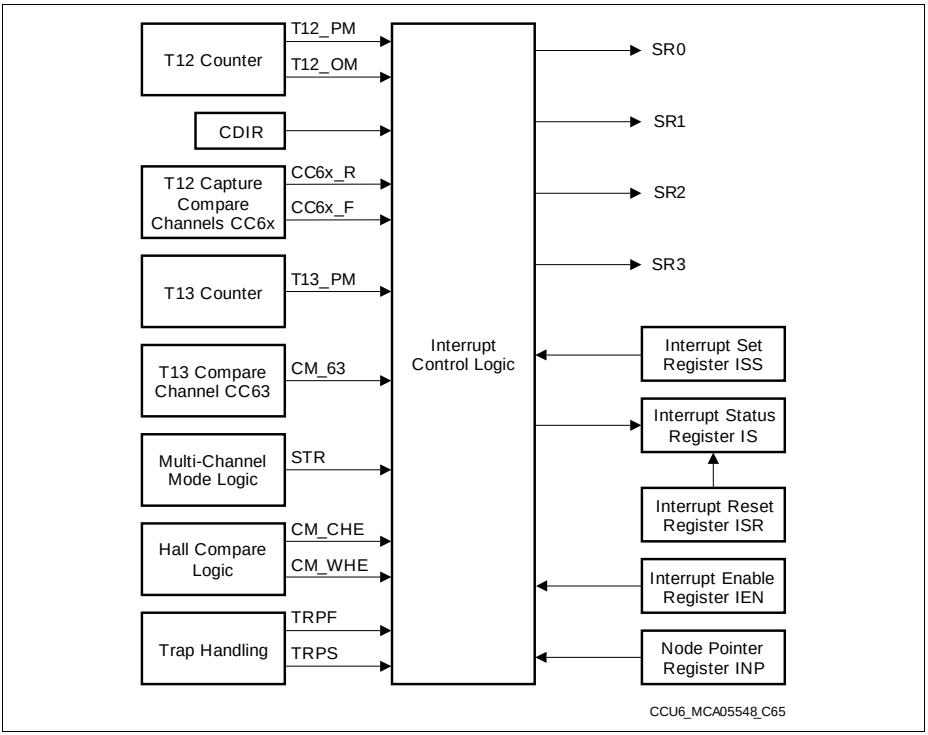


Figure 25-43 Interrupt Sources and Events

25.9.2 Interrupt Registers

25.9.2.1 Interrupt Status Register

Register IS contains the individual interrupt request bits. This register can only be read, write actions have no impact on the contents of this register. The SW can set or clear the bits individually by writing to the registers ISS (to set the bits) or to register ISR (to clear the bits).

The interrupt generation is independent from the value of the bits in register IS, e.g. the interrupt will be generated (if enabled) even if the corresponding bit is already set. The trigger for an interrupt generation is the detection of a set condition (by HW or SW) for the corresponding bit in register IS.

In compare mode (and hall mode), the timer-related interrupts are only generated while the timer is running (T1xR=1). In capture mode, the capture interrupts are also generated while the timer T12 is stopped.

Note: Not all bits in register IS can generate an interrupt. Other status bits have been added, that have a similar structure for their set and clear actions. It is recommended that SW checks the interrupt bits bit-wisely (instead of common OR over the bits).

IS
Interrupt Status Register **($A0_H$)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STR	IDLE	WHE	CHE	TRPS	TRPF	T13PM	T13CM	T12PM	T12OM	ICC62F	ICC62R	ICC61F	ICC61R	ICC60F	ICC60R
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
ICC60R, ICC61R, ICC62R	0, 2, 4	rh	**Capture, Compare-Match Rising Edge Flag** This bit indicates that event CC6x_R has been detected. This event occurs in compare mode when a compare-match is detected while T12 is counting up (CM_6x and CDIR = 0) and in capture mode when a rising edge is detected at the related input CC6xIN. 0_B The event has not yet been detected. 1_B The event has been detected.
ICC60F, ICC61F, ICC62F	1, 3, 5	rh	**Capture, Compare-Match Falling Edge Flag** This bit indicates that event CC6x_F has been detected. This event occurs in compare mode when a compare-match is detected while T12 is counting down (CM_6x and CDIR = 1) and in capture mode when a falling edge is detected at the related input CC6xIN. 0_B The event has not yet been detected. 1_B The event has been detected.
T12OM	6	rh	**Timer T12 One-Match Flag** This bit indicates that a timer T12 one-match while counting down (T12_OM and CDIR = 1) has been detected. 0_B The event has not yet been detected. 1_B The event has been detected.
T12PM	7	rh	**Timer T12 Period-Match Flag** This bit indicates that a timer T12 period-match while counting up (T12_PM and CDIR = 0) has been detected. 0_B The event has not yet been detected. 1_B The event has been detected.
T13CM	8	rh	**Timer T13 Compare-Match Flag** This bit indicates that a timer T13 compare-match (CM_63) has been detected. 0_B The event has not yet been detected. 1_B The event has been detected.
T13PM	9	rh	**Timer T13 Period-Match Flag** This bit indicates that a timer T13 period-match (T13_PM) has been detected. 0_B The event has not yet been detected. 1_B The event has been detected.

Field	Bits	Type	Description
TRPF	10	rh	**Trap Flag** This bit indicates if a trap condition (input $\overline{\text{CTRAP}}$ = 0 or by SW) is / has been detected. If TRPM2= 0, it becomes cleared automatically if $\overline{\text{CTRAP}}$ = 1 or TRPPEN = 0, whereas if TRPM2 = 1, it has to be cleared by writing RTRPF = 1. 0_B The trap condition has not been detected. 1_B The trap condition is / has been detected.
TRPS	11	rh	**Trap State**[1)] This bit indicates the actual trap state. It is set if TRPF = 1 and becomes cleared according to the mode selected in register TRPCTR. 0_B The trap state is not active. 1_B The trap state is active.
CHE	12	rh	**Correct Hall Event** This bit indicates that a correct Hall event (CM_CHE) has been detected. 0_B The event has not yet been detected. 1_B The event has been detected.
WHE	13	rh	**Wrong Hall Event** This bit indicates that a wrong Hall event (CM_WHE) has been detected. 0_B The event has not yet been detected. 1_B The event has been detected.
IDLE	14	rh	**IDLE State** If enabled by ENIDLE = 1, this bit is set together with bit WHE and it has to be cleared by SW. 0_B No action. 1_B Bit field MCMP is cleared, the selected outputs are set to passive state.
STR	15	rh	**Multi-Channel Mode Shadow Transfer Request** This bit indicates that a shadow transfer from MCMPS to MCMP (MCM_ST) has taken place. 0_B The event has not yet been detected. 1_B The event has been detected.
0	[31:16]	r	**Reserved;** Returns 0 if read; should be written with 0.

1) During the trap state, the selected outputs are set to the passive state. The logic level driven during the passive state is defined by the corresponding bit in register PSLR. Bits TRPS=1 and TRPF=0 can occur if the trap condition is no longer active but the selected synchronization has not yet taken place.

25.9.2.2 Interrupt Status Set Register

Register ISS contains individual interrupt request set bits to generate a CCU6 interrupt request by software. Writing a 1 sets the bit(s) in register IS at the corresponding bit position(s) and can generate an interrupt event (if available and enabled).
All bit positions read as 0.

ISS
Interrupt Status Set Register **($A4_H$)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S STR	**S IDLE**	**S WHE**	**S CHE**	**S WHC**	**S TRPF**	**S T13PM**	**S T13CM**	**S T12PM**	**S T12OM**	**S CC62F**	**S CC62R**	**S CC61F**	**S CC61R**	**S CC60F**	**S CC60R**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
SCC60R, SCC61R, SCC62R	0, 2, 4	w	**Set Capture, Compare-Match Rising Edge Flag** 0_B No action 1_B Bit CC6xR will be set.
SCC60F, SCC61F, SCC62F	1, 3, 5	w	**Set Capture, Compare-Match Falling Edge Flag** 0_B No action 1_B Bit CC6xF will be set.
ST12OM	6	w	**Set Timer T12 One-Match Flag** 0_B No action 1_B Bit T12OM will be set.
ST12PM	7	w	**Set Timer T12 Period-Match Flag** 0_B No action 1_B Bit T12PM will be set.
ST13CM	8	w	**Set Timer T13 Compare-Match Flag** 0_B No action 1_B Bit T13CM will be set.

Field	Bits	Type	Description
ST13PM	9	w	**Set Timer T13 Period-Match Flag** 0_B No action 1_B Bit T13PM will be set.
STRPF	10	w	**Set Trap Flag** 0_B No action 1_B Bits TRPF and TRPS will be set.
SWHC	11	w	**Software Hall Compare** 0_B No action 1_B The Hall compare action is triggered.
SCHE	12	w	**Set Correct Hall Event Flag** 0_B No action 1_B Bit CHE will be set.
SWHE	13	w	**Set Wrong Hall Event Flag** 0_B No action 1_B Bit WHE will be set.
SIDLE	14	w	**Set IDLE Flag** 0_B No action 1_B Bit IDLE will be set.
SSTR	15	w	**Set STR Flag** 0_B No action 1_B Bit STR will be set.
0	[31:16]	r	**Reserved;** Returns 0 if read; should be written with 0.

25.9.2.3 Status Reset Register

Register ISR contains bits to individually clear the interrupt event flags by software. Writing a 1 clears the bit(s) in register IS at the corresponding bit position(s).
All bit positions read as 0.

ISR
Interrupt Status Reset Register **($A8_H$)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R STR	R IDLE	R WHE	R CHE	0	R TRPF	R T13PM	R T13CM	R T12PM	R T12OM	R CC62F	R CC62R	R CC61F	R CC61R	R CC60F	R CC60R
w	w	w	w	r	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
RCC60R, RCC61R, RCC62R	0, 2, 4	w	**Reset Capture, Compare-Match Rising Edge Flag** 0_B No action 1_B Bit CC6xR will be cleared.
RCC60F, RCC61F, RCC62F	1, 3, 5	w	**Reset Capture, Compare-Match Falling Edge Flag** 0_B No action 1_B Bit CC6xF will be cleared.
RT12OM	6	w	**Reset Timer T12 One-Match Flag** 0_B No action 1_B Bit T12OM will be cleared.
RT12PM	7	w	**Reset Timer T12 Period-Match Flag** 0_B No action 1_B Bit T12PM IS will be cleared.
RT13CM	8	w	**Reset Timer T13 Compare-Match Flag** 0_B No action 1_B Bit T13CM will be cleared.
RT13PM	9	w	**Reset Timer T13 Period-Match Flag** 0_B No action 1_B Bit T13PM will be cleared.

Field	Bits	Type	Description
RTRPF	10	w	**Reset Trap Flag** 0_B No action 1_B Bit TRPF will be cleared (not taken into account while input $\overline{\text{CTRAP}}$=0 and TRPPEN=1.
RCHE	12	w	**Reset Correct Hall Event Flag** 0_B No action 1_B Bit CHE will be cleared.
RWHE	13	w	**Reset Wrong Hall Event Flag** 1_B No action 0_B Bit WHE will be cleared.
RIDLE	14	w	**Reset IDLE Flag** 0_B No action 1_B Bit IDLE will be cleared.
RSTR	15	w	**Reset STR Flag** 0_B No action 1_B Bit STR will be cleared.
0	11, [31:16]	r	**Reserved;** Returns 0 if read; should be written with 0.

25.9.2.4 Interrupt Enable Register

Register IEN contains the interrupt enable bits and a control bit to enable the automatic idle function in the case of a wrong hall pattern.

IEN
Interrupt Enable Register **($B0_H$)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENSTR	ENIDLE	ENWHE	ENCHE	0	ENTRPF	ENT13PM	ENT13CM	ENT12PM	ENT12OM	ENCC62F	ENCC62R	ENCC61F	ENCC61R	ENCC60F	ENCC60R
rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENCC60R, ENCC61R, ENCC62R	0, 2, 4	rw	**Capture, Compare-Match Rising Edge Interrupt Enable for Channel CC6x** 0_B No interrupt will be generated if the set condition for bit CC6xR in register IS occurs. 1_B An interrupt will be generated if the set condition for bit CC6xR in register IS occurs. The service request output that will be activated is selected by bit field INPCC6x.
ENCC60F, ENCC61F, ENCC62F	1, 3, 5	rw	**Capture, Compare-Match Falling Edge Interrupt Enable for Channel CC6x** 0_B No interrupt will be generated if the set condition for bit CC6xF in register IS occurs. 1_B An interrupt will be generated if the set condition for bit CC6xF in register IS occurs. The service request output that will be activated is selected by bit field INPCC6x.

Field	Bits	Type	Description
ENT12OM	6	rw	**Enable Interrupt for T12 One-Match** 0_B No interrupt will be generated if the set condition for bit T12OM in register IS occurs. 1_B An interrupt will be generated if the set condition for bit T12OM in register IS occurs. The service request output that will be activated is selected by bit field INPT12.
ENT12PM	7	rw	**Enable Interrupt for T12 Period-Match** 0_B No interrupt will be generated if the set condition for bit T12PM in register IS occurs. 1_B An interrupt will be generated if the set condition for bit T12PM in register IS occurs. The service request output that will be activated is selected by bit field INPT12.
ENT13CM	8	rw	**Enable Interrupt for T13 Compare-Match** 0_B No interrupt will be generated if the set condition for bit T13CM in register IS occurs. 1_B An interrupt will be generated if the set condition for bit T13CM in register IS occurs. The service request output that will be activated is selected by bit field INPT13.
ENT13PM	9	rw	**Enable Interrupt for T13 Period-Match** 0_B No interrupt will be generated if the set condition for bit T13PM in register IS occurs. 1_B An interrupt will be generated if the set condition for bit T13PM in register IS occurs. The service request output that will be activated is selected by bit field INPT13.
ENTRPF	10	rw	**Enable Interrupt for Trap Flag** 0_B No interrupt will be generated if the set condition for bit TRPF in register IS occurs. 1_B An interrupt will be generated if the set condition for bit TRPF in register IS occurs. The service request output that will be activated is selected by bit field INPERR.

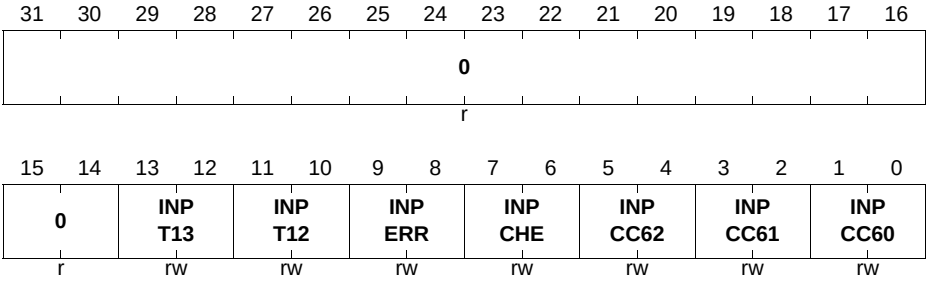
Field	Bits	Type	Description
ENCHE	12	rw	**Enable Interrupt for Correct Hall Event** 0_B No interrupt will be generated if the set condition for bit CHE in register IS occurs. 1_B An interrupt will be generated if the set condition for bit CHE in register IS occurs. The service request output that will be activated is selected by bit field INPCHE.
ENWHE	13	rw	**Enable Interrupt for Wrong Hall Event** 0_B No interrupt will be generated if the set condition for bit WHE in register IS occurs. 1_B An interrupt will be generated if the set condition for bit WHE in register IS occurs. The service request output that will be activated is selected by bit field INPERR.
ENIDLE	14	rw	**Enable Idle** This bit enables the automatic entering of the idle state (bit IDLE will be set) after a wrong hall event has been detected (bit WHE is set). During the idle state, the bit field MCMP is automatically cleared. 0_B The bit IDLE is not automatically set when a wrong hall event is detected. 1_B The bit IDLE is automatically set when a wrong hall event is detected.
ENSTR	15	rw	**Enable Multi-Channel Mode Shadow Transfer Interrupt** 0_B No interrupt will be generated if the set condition for bit STR in register IS occurs. 1_B An interrupt will be generated if the set condition for bit STR in register IS occurs. The service request output that will be activated is selected by bit field INPCHE.
0	11, [31:16]	r	**Reserved;** Returns 0 if read; should be written with 0.

25.9.2.5 Interrupt Node Pointer Register

Register INP contains the interrupt node pointers allowing a flexible interrupt handling. These bit fields define which service request output will be activated if the corresponding interrupt event occurs and the interrupt generation for this event is enabled.

INP

Interrupt Node Pointer Register **(AC_H)** **Reset Value: 0000 3940$_H$**



Field	Bits	Type	Description
INPCC60, INPCC61, INPCC62	[1:0], [3:2], [5:4]	rw	**Interrupt Node Pointer for Channel CC6x Interrupts** This bit field defines the service request output activated due to a set condition for bit CC6xR (if enabled by bit ENCC6xR) or for bit CC6xF (if enabled by bit ENCC6xF). 00_B Service request output SR0 is selected. 01_B Service request output SR1 is selected. 10_B Service request output SR2 is selected. 11_B Service request output SR3 is selected.
INPCHE	[7:6]	rw	**Interrupt Node Pointer for the CHE Interrupt** This bit field defines the service request output activated due to a set condition for bit CHE (if enabled by bit ENCHE) of for bit STR (if enabled by bit ENSTR). Coding see INPCC6x.
INPERR	[9:8]	rw	**Interrupt Node Pointer for Error Interrupts** This bit field defines the service request output activated due to a set condition for bit TRPF (if enabled by bit ENTRPF) or for bit WHE (if enabled by bit ENWHE). Coding see INPCC6x.
INPT12	[11:10]	rw	**Interrupt Node Pointer for Timer12 Interrupts** This bit field defines the service request output activated due to a set condition for bit T12OM (if enabled by bit ENT12OM) or for bit T12PM (if enabled by bit ENT12PM). Coding see INPCC6x.

Field	Bits	Type	Description
INPT13	[13:12]	rw	**Interrupt Node Pointer for Timer13 Interrupt** This bit field defines the service request output activated due to a set condition for bit T13CM (if enabled by bit ENT13CM) or for bit T13PM (if enabled by bit ENT13PM). Coding see INPCC6x.
0	[31:14]	r	**Reserved;** Returns 0 if read; should be written with 0.

25.10 General Module Operation

This section provides information about the:

- Input Selection (see **Section 25.10.1**)
- Input Monitoring (see **Section 25.10.2**)
- OCDS Suspend Functionality (see **Section 25.10.3**)
- OCDS Trigger Bus (OTGB) Interface (see **Section 25.10.4**)
- General Register Description (see **Section 25.10.5**)
- BPI Register Description (see **Section 25.10.6**)

25.10.1 Input Selection

Each CCU6 input signal can be selected from a vector of four or eight possible inputs by programming the port input select registers **PISEL0** and **PISEL2**. This permits to adapt the pin functionality of the device to the application requirements.
The output pins for the module output signals are chosen in the ports.

Naming convention:
The input vector CC60IN[D:A] for input signal CC60IN is composed of the signals CC60INA to CC60IND.

Note: All functional inputs of the CCU6 are synchronized to f_{CC6} before they affect the module internal logic. The resulting delay of $2/f_{CC6}$ and for asynchronous signals an additional uncertainty of $1/f_{CC6}$ have to be taken into account for precise timing calculation. An edge of an input signal can only be correctly detected if the high phase and the low phase of the input signal are both longer than $1/f_{CC6}$.

25.10.2 Input Monitoring

A selected event which occurs at each CCU6 input signal can be monitored through **IMON**.x. Every input signal can be included for the detection of a lost bit event (**IMON**.LBE) if enabled through its individual lost indicator enable bits (**LI**.yEN). The lost bit event occurs if a selected event occurs again with the previous event captured (IMON.x remains set) and its lost indicator is enabled for at least one of the monitored input signals. The lost bit event can be enabled (**LI**.LBEEN) for an interrupt to be generated at one of the SRx line, selected through **LI**.INPLBE. The LBE output signal of the kernel can be connected to a capture input to indicate when does the lost bit event happens.

The lost bit event can be used as a kind of interrupt or event watchdog to monitor if an action related to an event has been processed before a second event of the same type occurs. Like this, if a certain event is treated by an interrupt that should be monitored, the related indication flag has to be cleared by SW. If the SW has not yet cleared the flag and the event occurs again, the event is considered as being lost and another interrupt can be generated to inform the system about the loss. This can be also used to indicate that input events occur too often and the main task has not enough time to treat them.

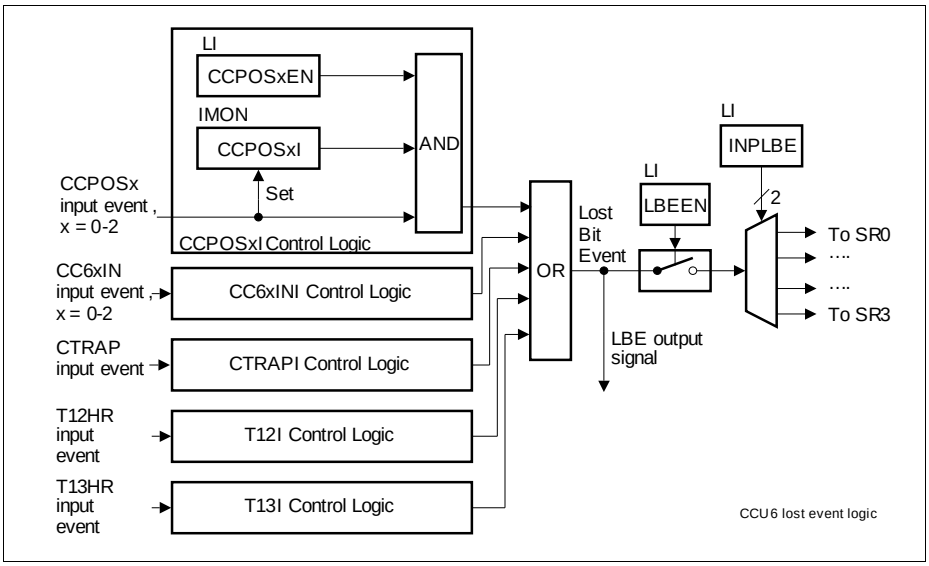


Figure 25-44 Lost Event Logic

25.10.3 OCDS Suspend

The behavior of CCU6 upon an OCDS suspend request is controlled by the **OCS** register. CCU6 supports both Hard Suspend Mode and Soft Suspend Mode.

Hard Suspend Mode

In Hard Suspend Mode the CCU6 kernel clock is switched off immediately. Reading and writing of registers is possible but will enable the kernel clock for a few cycles.

Attention: Register accesses with clocking in Hard Suspend Mode can have unintended side effects like signals becoming and staying active. This can affect also other modules, so a CCU6 kernel reset might not be sufficient to bring the system into a defined state.

Soft Suspend Mode

In Soft Suspend Mode CCU6 may finalize specific actions before it enters the suspended state with **OCS**.SUSSTA set. This can be advantageous for applications (e.g. motor control) where critical system states could occur if the kernel clock would be switched off immediately upon an OCDS suspend request.

Soft Suspend mode does not influence the kernel clock. Reading and writing of registers is possible without the side effects that may occur when reading/writing registers in Hard Suspend Mode.

For CCU6, two basic soft suspend options (**Stop Mode 0**, **Stop Mode 1**) are available, selected via bit field **OCS**.SUS.

In addition, the internal functional blocks (T12, T13, Hall logic, Trap logic) may individually be programmed via their Sensitivity Bits in register **KSCSR** to accept or ignore a suspend request. If the request sensitivity is disabled, the block continues normal operation. If the request sensitivity is enabled, the block operates as specified for the selected stop mode. The mapping of the CCU6 functional blocks to their Sensitivity Bits is shown in **Table 25-12**.

Stop Mode 0

In Stop Mode 0, if selected to be stopped, the Hall and Trap logic stops immediately, while Timer T12 and/or T13 continues normal operation (if running) until it reaches the end of the PWM period; then it stops (same stop condition as in single shot mode). When Timer T12 stops, the capture inputs CC6xIN are frozen.

Stop Mode 1

In Stop Mode 1, if selected to be stopped, the internal functional blocks (T12, T13, Hall logic, Trap logic) will stop immediately upon a suspend request.

If the sensitivity bits for Timer T12 or T13 are set, the corresponding output lines enabled for the trap condition are set to their passive values (similar to a trap condition).

Table 25-12 summarizes the reaction of the CCU6 functional blocks to OCDS suspend requests.

Table 25-12 CCU6 Functional Blocks

Block	Reaction to OCDS Suspend Request	Sensitivity Bit
0	**Timer T12:** if bit SB0 = 1_B, - in **Stop Mode 0** Timer T12 continues (if running) until the end of the PWM period. Then it stops and the CCxIN input stages are frozen. - in **Stop Mode 1** Timer T12 stops immediately and the CC6xIN input stages are frozen. Output lines CC6x, COUT6x with enabled trap functionality are set to their passive states.	KSCSR.SB0
1	**Timer T13:** if bit SB1 = 1_B, - in **Stop Mode 0** Timer T13 continues (if running) until the end of the PWM period. Then it stops. - in **Stop Mode 1** Timer T13 stops immediately. If trap functionality for COUT63 is enabled, it is set to the passive state.	KSCSR.SB1
2	**Hall Logic:** if bit SB2 = 1_B, the hall logic is stopped immediately and the CCPOSx input stages are frozen (same behavior for Stop Mode 0 and 1).	KSCSR.SB2
3	**Trap Logic:** if bit SB3 = 1_B, the trap logic is stopped immediately and the CTRAP input stage is frozen (same behavior for Stop Mode 0 and 1).	KSCSR.SB3

25.10.4 OCDS Trigger Bus (OTGB) Interface

The CCU6 kernel provides a set of 16 internal status signals that are combined to a Trigger Set. The CCU6 Trigger Set is shown in **Table 25-13**. It is output on OTGB0 or OTGB1 controlled by the **OCS** register.

Table 25-13 TS16_CCU6 Trigger Set CCU6

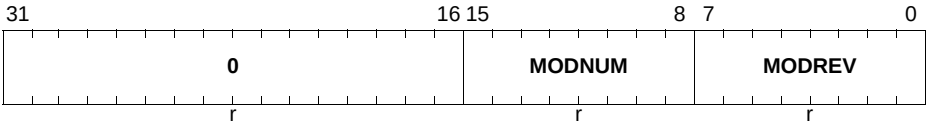
Bits	Name	Description
0	T12pre	T12 prescaler output
1	T13pre	T13 prescaler output
2	T12_cm0	T12 compare match channel 0
3	T12_cm1	T12 compare match channel 1
4	T12_cm2	T12 compare match channel 2
5	T12_pm	T12 period match
6	T12_om	T12 one match
7	T12_cdir	T12 count direction
8	T12_zm	T12 zero match
9	T13_cm	T13 compare match
10	T13_pm	T13 period match
11	T13_zm	T13 zero match
12	mcm_st	MCM shadow transfer
13	che	Correct Hall event
14	whe	Wrong Hall event
15	trpf	Trap flag

25.10.5 General Registers

25.10.5.1 ID Register

The CCU6 Module Identification Register ID contains read-only information about the module identification number and its revision.

ID
Module Identification Register **(08_H)** **Reset Value: 0000 54XX$_H$**



Field	Bits	Type	Description
MODREV	[7:0]	r	**Module Revision Number** MODREV defines the module revision number. The value of a module revision starts with 01_H (first revision), 02_H, 03_H, ... up to FF_H.
MODNUM	[15:8]	r	**Module Number Value** This bit field defines the module identification number for the CCU6: 54_H
0	[31:16]	r	**Reserved** Read as 0.

25.10.5.2 Port Input Select Registers

Registers PISEL0 and PISEL2 contain bit fields selecting the actual input signal for the module inputs.

PISEL0
Port Input Select Register 0 **(10_H)** **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IST12HR		ISPOS2		ISPOS1		ISPOS0		ISTRP		ISCC62		ISCC61		ISCC60	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
ISCC60	[1:0]	rw	**Input Select for CC60** This bit field defines the input signal used as CC60 capture input. 00_B The signal CC60INA is selected. 01_B The signal CC60INB is selected. 10_B The signal CC60INC is selected. 11_B The signal CC60IND is selected.
ISCC61	[3:2]	rw	**Input Select for CC61** This bit field defines the input signal used as CC61 capture input. 00_B The signal CC61INA is selected. 01_B The signal CC61INB is selected. 10_B The signal CC61INC is selected. 11_B The signal CC61IND is selected.
ISCC62	[5:4]	rw	**Input Select for CC62** This bit field defines the input signal used as CC62 capture input. 00_B The signal CC62INA is selected. 01_B The signal CC62INB is selected. 10_B The signal CC62INC is selected. 11_B The signal CC62IND is selected.

Field	Bits	Type	Description
ISTRP	[7:6]	rw	**Input Select for CTRAP** This bit field defines the input signal used as CTRAP input. 00_B The signal CTRAPA is selected. 01_B The signal CTRAPB is selected. 10_B The signal CTRAPC is selected. 11_B The signal CTRAPD is selected.
ISPOS0	[9:8]	rw	**Input Select for CCPOS0** This bit field defines the input signal used as CCPOS0 input. 00_B The signal CCPOS0A is selected. 01_B The signal CCPOS0B is selected. 10_B The signal CCPOS0C is selected. 11_B The signal CCPOS0D is selected.
ISPOS1	[11:10]	rw	**Input Select for CCPOS1** This bit field defines the input signal used as CCPOS1 input. 00_B The signal CCPOS1A is selected. 01_B The signal CCPOS1B is selected. 10_B The signal CCPOS1C is selected. 11_B The signal CCPOS1D is selected.
ISPOS2	[13:12]	rw	**Input Select for CCPOS2** This bit field defines the input signal used as CCPOS2 input. 00_B The signal CCPOS2A is selected. 01_B The signal CCPOS2B is selected. 10_B The signal CCPOS2C is selected. 11_B The signal CCPOS2D is selected.

Field	Bits	Type	Description
IST12HR	[15:14]	rw	**Input Select for T12HR** This bit field defines the input signal used as T12HR input. 00_B Either signal T12HRA (if T12EXT = 0) or T12HRE (if T12EXT = 1) is selected. 01_B Either signal T12HRB (if T12EXT = 0) or T12HRF (if T12EXT = 1) is selected. 10_B Either signal T12HRC (if T12EXT = 0) or T12HRG (if T12EXT = 1) is selected. 11_B Either signal T12HRD (if T12EXT = 0) or T12HRH (if T12EXT = 1) is selected.
0	[31:16]	r	**Reserved** Returns 0 if read, should be written with 0.

PISEL2

Port Input Select Register 2 **(14_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								**T13 EXT**	**T12 EXT**	**ISCNT13**		**ISCNT12**		**IST13HR**	
r								rw	rw	rw		rw		rw	

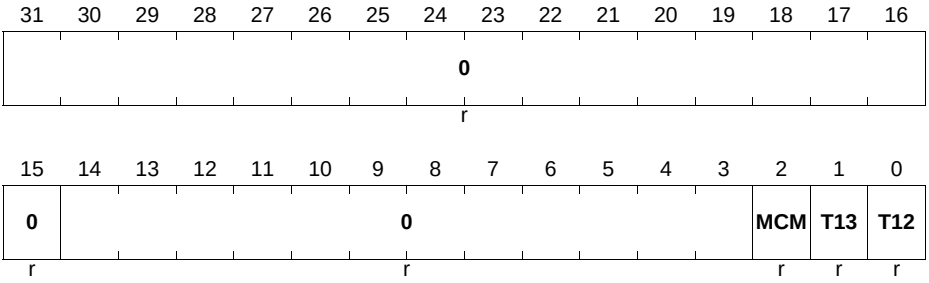
Field	Bits	Type	Description
IST13HR	[1:0]	rw	**Input Select for T13HR** This bit field defines the input signal used as T13HR input. 00_B Either signal T13HRA (if T13EXT = 0) or T13HRE (if T13EXT = 1) is selected. 01_B Either signal T13HRB (if T13EXT = 0) or T13HRF (if T13EXT = 1) is selected. 10_B Either signal T13HRC (if T13EXT = 0) or T13HRG (if T13EXT = 1) is selected. 11_B Either signal T13HRD (if T13EXT = 0) or T13HRH (if T13EXT = 1) is selected.
ISCNT12	[3:2]	rw	**Input Select for T12 Counting Input** This bit field defines the input event leading to a counting action of T12. 00_B The T12 prescaler generates the counting events. Bit TCTR4.T12CNT is not taken into account. 01_B Bit TCTR4.T12CNT written with 1 is a counting event. The T12 prescaler is not taken into account. 10_B The timer T12 is counting each rising edge detected in the selected T12HR signal. 11_B The timer T12 is counting each falling edge detected in the selected T12HR signal.
ISCNT13	[5:4]	rw	**Input Select for T13 Counting Input** This bit field defines the input event leading to a counting action of T13. 00_B The T13 prescaler generates the counting events. Bit TCTR4.T13CNT is not taken into account. 01_B Bit TCTR4.T13CNT written with 1 is a counting event. The T13 prescaler is not taken into account. 10_B The timer T13 is counting each rising edge detected in the selected T13HR signal. 11_B The timer T13 is counting each falling edge detected in the selected T13HR signal.
T12EXT	6	rw	**Extension for T12HR Inputs** This bit extends the 2-bit field IST12HR. 0_B One of the signals T12HR[D:A] is selected. 1_B One of the signals T12HR[H:E] is selected.

Field	Bits	Type	Description
T13EXT	7	rw	**Extension for T13HR Inputs** This bit extends the 2-bit field IST13HR. 0_B One of the signals T13HR[D:A] is selected. 1_B One of the signals T13HR[H:E] is selected.
0	[31:8]	r	**Reserved;** Returns 0 if read; should be written with 0.

25.10.5.3 Module Configuration Register

The module configuration register contains bits describing the functionality that is available in the CCU6 module.

MCFG
Module Configuration Register (04_H) Reset Value: 0000 0007$_H$



Field	Bits	Type	Description
T12	0	r	**T12 Available** This bit indicates if the T12 block is available. 0_B The T12 block is not available. A write access to T12PR is ignored. 1_B The T12 block is available. A write access to T12PR is executed.
T13	1	r	**T13 Available** This bit indicates if the T13 block is available. 0_B The T13 block is not available. A write access to T13PR is ignored. 1_B The T13 block is available. A write access to T13PR is executed.
MCM	2	r	**Multi-Channel Mode Available** This bit indicates if the multi-channel mode functionality is available. 0_B The multi-channel mode functionality is not available. A write access to MCMOUTS is ignored. 1_B The multi-channel mode functionality is available. A write access to MCMOUTS is executed.
0	[31:3]	r	**Reserved;** read as 0; should be written with 0.

25.10.5.4 Input Monitoring Register

The input monitoring register monitors the occurrence of a selected event for the input signals. If a hardware event triggers the setting of bit IMON.x and the same bit is written with 1 via software at the same time, then the corresponding bit is cleared (software overrules hardware). The lost bit event is indicated if an event is detected again at one or more input signals with its lost indicator enabled.

Note: The register is only applicable in capture modes if the edges are selected through T12MSEL.MSEL6x.

IMON
Input Monitoring Register **(98_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						T13HRI	T12HRI	CTRAPI	CC62INI	CC61INI	CC60INI	CCPOS2I	CCPOS1I	CCPOS0I	LBE
r						rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
LBE	0	rwh	**Lost Bit Event** This bit determines if a lost bit event has occurred. A lost bit event occurs when a selected event occurs again with the previous event captured (IMON.x remains set) and its lost indicator is enabled, for at least one of the monitored input signals. The bit can be cleared by writing a 1 to the same bit position, while writing a 0 has no effect. 0_B The lost bit event has not occurred. 1_B The lost bit event has occurred.
CCPOSxI (x = 0 -2)	x + 1	rwh	**Event indication for input signal CCPOSx** The bit determines if the selected event has occurred via an edge detection. The bit can be cleared by writing a 1 to the same bit position, while writing a 0 has no effect. 0_B A selected event has not occurred. 1_B Edge detection indicates a selected event has occurred. *Note: The dedicated edge is indicated for a selected event if Hysteretic-like Control or Capture modes are initialized in T12MSEL.MSEL6x. If these modes are not selected, then all edges will be indicated as an event for the inputs.*
CC6xINI (x = 0 -2)	x + 4	rwh	**Event indication for input signal CC6xIN** The bit determines if the selected event has occurred via an edge detection. The bit can be cleared by writing a 1 to the same bit position, while writing a 0 has no effect. 0_B A selected event has not occurred. 1_B Edge detection indicates a selected event has occurred.
CTRAPI	7	rwh	**Event indication for input signal CTRAP** The bit determines if the selected event has occurred via an edge detection. The bit can be cleared by writing a 1 to the same bit position, while writing a 0 has no effect. 0_B An event has not occurred. 1_B Edge detection indicates an event has occurred.

Field	Bits	Type	Description
T12HRI	8	rwh	**Event indication for input signal T12HR** The bit determines if the selected event has occurred via an edge detection. The bit can be cleared by writing a 1 to the same bit position, while writing a 0 has no effect. 0_B An event has not occurred. 1_B Edge detection indicates an event has occurred.
T13HRI	9	rwh	**Event indication for input signal T13HR** The bit determines if the selected event has occurred via an edge detection. The bit can be cleared by writing a 1 to the same bit position, while writing a 0 has no effect. 0_B An event has not occurred. 1_B Edge detection indicates an event has occurred.
0	[31:10]	r	**Reserved;** Returns 0 if read; should be written with 0.

25.10.5.5 Lost Indicator Register

The lost indicator register has the lost indicator enable bits for its detected event at the input signals. The lost bit event can then be enabled as an output signal through one of the service request lines.

LI
Lost Indicator Register **($9C_H$)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INPLBE		LBEEN	0			T13HREN	T12HREN	CTRAPEN	CC62INEN	CC61INEN	CC60INEN	CCPOS2EN	CCPOS1EN	CCPOS0EN	0
rw		rw	r			rw	rw	rw	rw	rw	rw	rw	rw	rw	r

Field	Bits	Type	Description
CCPOSxEN (x = 0 -2)	x + 1	rw	**Lost Indicator Enable for input signal CCPOSx** This bit determines if the monitored event at the input signal is enabled for the detection of a lost bit event. 0_B Input signal is disabled for a lost bit event detection. 1_B Input signal is enabled for a lost bit event detection.
CC6xINEN (x = 0 -2)	x + 4	rw	**Lost Indicator Enable for input signal CC6xIN** This bit determines if the monitored event at the input signal is enabled for the detection of a lost bit event. 0_B Input signal is disabled for a lost bit event detection. 1_B Input signal is enabled for a lost bit event detection.

Field	Bits	Type	Description
CTRAPEN	7	rw	**Lost Indicator Enable for input signal CTRAP** This bit determines if the monitored event at the input signal is enabled for the detection of a lost bit event. 0_B Input signal is disabled for a lost bit event detection. 1_B Input signal is enabled for a lost bit event detection.
T12HREN	8	rw	**Lost Indicator Enable for input signal T12HR** This bit determines if the monitored event at the input signal is enabled for the detection of a lost bit event. 0_B Input signal is disabled for a lost bit event detection. 1_B Input signal is enabled for a lost bit event detection.
T13HREN	9	rw	**Lost Indicator Enable for input signal T13HR** This bit determines if the monitored event at the input signal is enabled for the detection of a lost bit event. 0_B Input signal is disabled for a lost bit event detection. 1_B Input signal is enabled for a lost bit event detection.
LBEEN	13	rw	**Interrupt Enable for Lost Bit Event** This bit determines if a SRx line is activated if lost bit event is detected. 0_B Lost bit event is disabled for the activation of a SRx line. 1_B Lost bit event is enabled for the activation of a SRx line.
INPLBE	[15:14]	rw	**Interrupt Node Pointer for lost bit event** This bit field defines which service request output line is selected to output an lost event alert for an enabled lost bit event. 00_B Service request output SR0 is selected. 01_B Service request output SR1 is selected. 10_B Service request output SR2 is selected. 11_B Service request output SR3 is selected.
0	0, [12:10], [31:16]	r	**Reserved;** Returns 0 if read; should be written with 0.

25.10.5.6 Kernel State Control Sensitivity Register

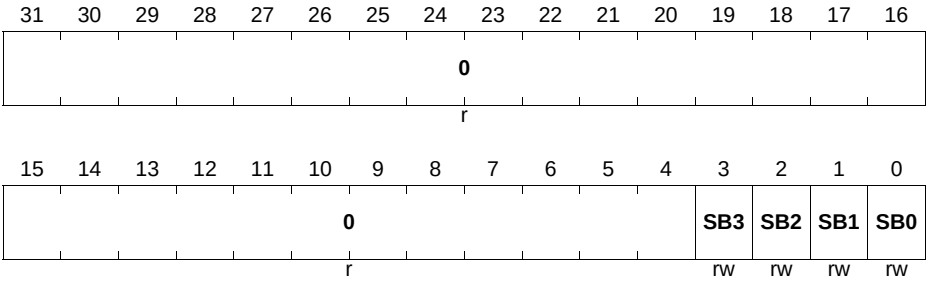
The kernel state control sensitivity register bits define which internal block is affected by Stop Modes 0 and 1, as described in section **Soft Suspend Mode**.

The Kernel State Control Sensitivity Register (KSCSR) is cleared by Debug Reset.

The register can only be written while the OCDS is enabled (OCDS enable = ´1´).

KSCSR
Kernel State Control Sensitivity Register

(1C$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
SB0, **SB1,** **SB2,** **SB3**	0, 1, 2, 3	rw	**Sensitivity Block x** This bit defines if block x of the CCU6 kernel is sensitive to Stop Mode 0 or Stop Mode 1. The functional definition of the blocks is given in **Table 25-12**. 0_B Block x is not sensitive to Stop Mode 0 or Stop Mode 1. It continues normal operation without respecting the defined stop condition. 1_B Block x is sensitive to Stop Mode 0 or Stop Mode 1. It is respecting the defined stop condition.
0	[31:4]	r	**Reserved;** Returns 0 if read; should be written with 0.

25.10.6 BPI Registers

This section describes the registers of the BPI (Bus Peripheral Interface). **Figure 25-45** shows all registers associated with the BPI for one CCU6 kernel.

BPI Registers Overview

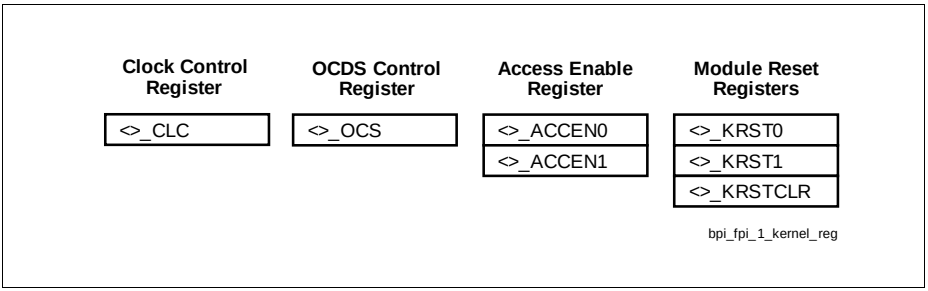


Figure 25-45 BPI Registers

Table 25-14 gives an overview of the BPI registers, including their access modes and reset class.

Table 25-14 Registers Overview - BPI Registers

Register Short Name	Description	Offset Addr.	Access Mode		Reset	Page Num.
			Read	Write		
CLC	Clock Control Register	00_H	U, SV	SV, E, P	Application Reset	25-130
-	Kernel Registers	-	-	-	-	-
OCS	OCDS Control and Status Register	$E8_H$	U, SV	SV, P	Debug Reset	25-131
KRSTCLR	Reset Status Clear Register	EC_H	U, SV	SV, E, P	Application Reset	25-137
KRST1	Reset Control Register 1	$F0_H$	U, SV	SV, E, P	Application Reset	25-136
KRST0	Reset Control Register 0	$F4_H$	U, SV	SV, E, P	Application Reset	25-135

Table 25-14 Registers Overview - BPI Registers

Register Short Name	Description	Offset Addr.	Access Mode		Reset	Page Num.
			Read	Write		
ACCEN1	Access Enable Register 1	$F8_H$	U, SV	SV, SE	Application Reset	25-134
ACCEN0	Access Enable Register 0	FC_H	U, SV	SV, SE	Application Reset	25-133

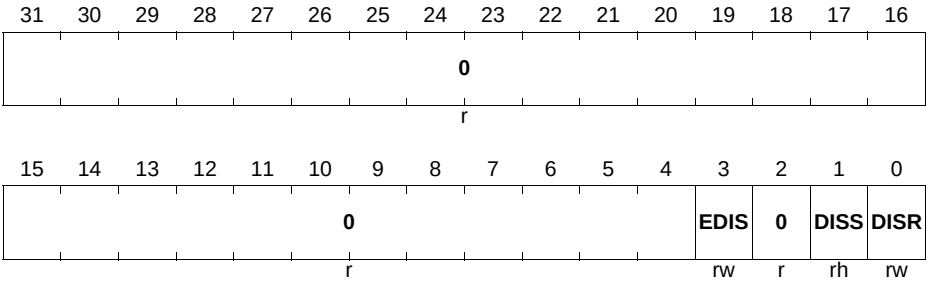
Note: Register bits marked “r” in the following register description are virtual registers and do not contain flip-flops. They are always read as 0.

25.10.6.1 System Registers

Clock Control Register (CLC)

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI for the CCU6. Where a module kernel is connected to the CLC clock control interface, CLC controls the f_{CCU6} module clock signal and Sleep Mode for the module.

CLC
Clock Control Register (00_H) Reset Value: $0000\ 0003_H$



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. 0_B Module disable is not requested. 1_B Module disable is requested.
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module. 0_B Module is enabled. 1_B Module is disabled.
EDIS	3	rw	**Sleep Mode Enable Control** Used to control module's sleep mode. 0_B Sleep Mode request is regarded. Module is enabled to go into Sleep Mode. 1_B Sleep Mode request is disregarded: Sleep Mode cannot be entered upon a request.

Field	Bits	Type	Description
0	[31:16], [15:4], 2	r	**Reserved** Read as 0; should be written with 0.

Note: Upon an accepted Sleep Mode request (with EDIS = '1'), or upon a disable request (DISR = '1'), the CCU6 kernel clock is switched off immediately. Therefore, software should ensure that the system controlled by the CCU6 kernel has reached a safe state before triggering a Sleep Mode or module disable request.

OCDS Control and Status Register (OCS)

The OCDS Control and Status (OCS) register is cleared by Debug Reset.

The OCS register can only be written when the OCDS is enabled. If OCDS is being disabled, the OCS register value will not change. When OCDS is disabled the OCS suspend control is ineffective.

Write access is 32 bit wide only and requires Supervisor Mode.

OCS
OCDS Control and Status Register ($E8_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**SUSSTA**	**SUS_P**	**SUS**				**0**							
r		rh	w	rw				r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												**TG_P**	**TGB**	**TGS**	
r												w	rw	rw	

Field	Bits	Type	Description
TGS	[1:0]	rw	**Trigger Set for OTGB0/1** 0_H No Trigger Set output 1_H Trigger Set TS16_CCU6 (**Table 25-13**) **others**, reserved
TGB	2	rw	**OTGB0/1 Bus Select** 0_B Trigger Set is output on OTGB0 1_B Trigger Set is output on OTGB1

Field	Bits	Type	Description
TG_P	3	w	**TGS, TGB Write Protection** TGS and TGB are only written when TG_P is 1, otherwise unchanged. Read as 0.
SUS	[27:24]	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0_H Will not suspend 1_H Hard suspend. Clock is switched off immediately. 2_H Soft suspend, Stop Mode 0 3_H Soft suspend, Stop Mode 1 **others**, reserved Effects of soft suspend options on CCU6 Functional Blocks are described in section **Soft Suspend Mode**
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	[23:4], [31:30]	r	**Reserved** Read as 0; must be written with 0.

Access Enable Register 0 (ACCEN0)

The Access Enable Register 0 controls write access[1)] for transactions with the on chip bus master TAG ID 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000_B, EN1 -> TAG ID 000001_B, ..., EN31 -> TAG ID 011111_B.

1) The BPI Access Enable functionality controls only write transactions to the CLC, OCS, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers.

ACCEN0
Access Enable Register 0 **(FC_H)** **Reset Value: FFFF FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN 31	**EN 30**	**EN 29**	**EN 28**	**EN 27**	**EN 26**	**EN 25**	**EN 24**	**EN 23**	**EN 22**	**EN 21**	**EN 20**	**EN 19**	**EN 18**	**EN 17**	**EN 16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN 15	**EN 14**	**EN 13**	**EN 12**	**EN 11**	**EN 10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

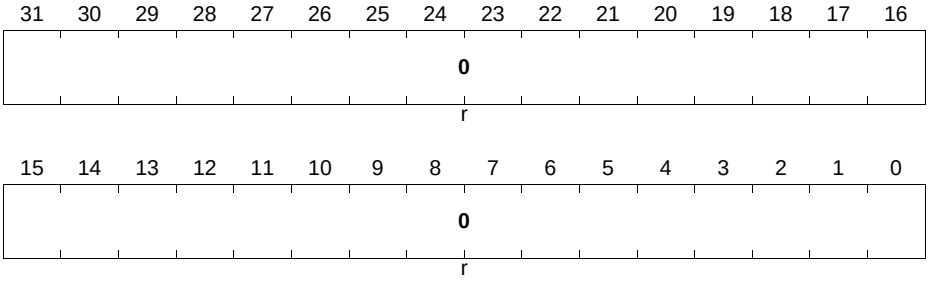
Field	Bits	Type	Description
ENn (n = 0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Access Enable Register 1 (ACCEN1)

The Access Enable Register 1 controls write access[1)] for transactions with the on chip bus master TAG ID 100000_B to 111111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI is prepared for a 6-bit TAG ID. ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the AURIX devices.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000_B, EN1 -> TAG ID 100001_B, ..., EN31 -> TAG ID 111111_B.

ACCEN1
Access Enable Register 1 **($F8_H$)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
0	[31:0]	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 0 (KRST0)

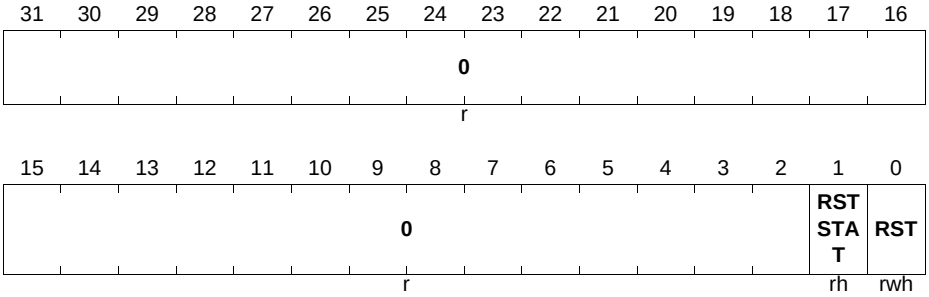
The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers related to the module kernel. The RST bit will be re-set (cleared to ‘0’) by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI in the same clock cycle the RST bit is re-set by the BPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the CLR bit in the related KRSTCLR register.

Note: During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge

*Note: A kernel reset, initiated via registers KRST0/KRST1, does not affect the port logic. Because register **PSLR** is set to its default value, the passive level 0 is selected for the outputs CC60..CC62 and COUT60..COUT63 upon a kernel reset. Therefore, software must ensure appropriate levels for the external system at the port pins in case they are different from the default values selected via PSLR.*

KRST0
Kernel Reset Register 0 **($F4_H$)** **Reset Value: 0000 0000$_H$**

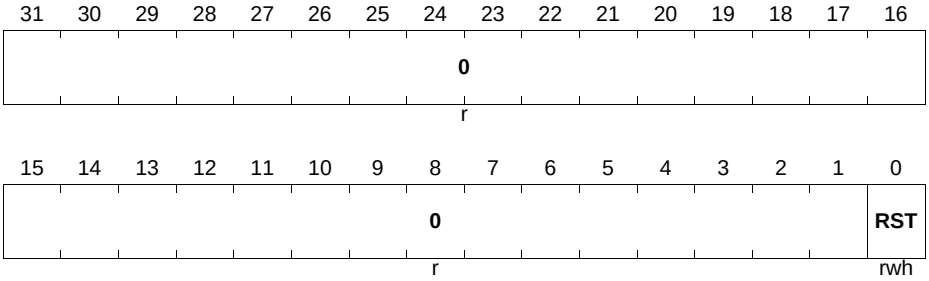


Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) by the BPI after the kernel reset was executed.
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI after the execution of a kernel reset in the same clock cycle both reset bits. 0_B No kernel reset was executed 1_B Kernel reset was executed This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register.
0	[31:2]	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1 (KRST1)

The Kernel Reset Register 1 is used to reset the related module kernel. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers (KRSTx1.RST and KRSTx0.RST) related to the module kernel that should be reset. The RST bit will be re-set (cleared to ‘0’) by the BPI with the end of the BPI kernel reset sequence.

KRST1
Kernel Reset Register 1 **($F0_H$)** **Reset Value: 0000 0000$_H$**



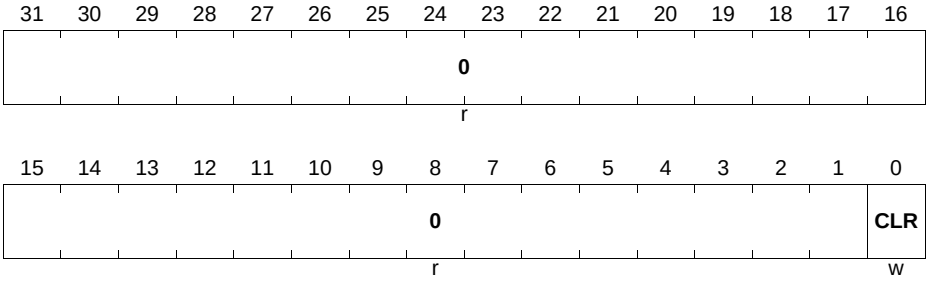
Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers are set. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) by the BPI after the kernel reset was executed.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Status Clear Register (KRSTCLR)

The Kernel Reset Status Clear Register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR

Kernel Reset Status Clear Register (EC$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT Read always as 0.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

25.11 Implementation

This chapter describes the implementation of the CCU6 module in the TC21x/TC22x/TC23x device.

- Address map (see **Section 25.11.1**)
- Module output select (see **Section 25.11.2**)
- Synchronous start (see **Section 25.11.3**)
- Digital Connections (see **Section 25.11.4**)

25.11.1 Address Map

There are two CCU6 kernels in the TC21x/TC22x/TC23x, namely CCU60 and CCU61. The CCU6061 module consists of CCU60 and CCU61 kernels .

Table 25-15 Registers Address Space

Module	Base Address	End Address	Note
CCU60	F000 2A00$_H$	F000 2AFF$_H$	CCU6061 module includes CCU60 and CCU61 kernels
CCU61	F000 2B00$_H$	F000 2BFF$_H$	CCU6061 module includes CCU60 and CCU61 kernels

Table 25-16 Registers Overview - CCU6 Module Registers

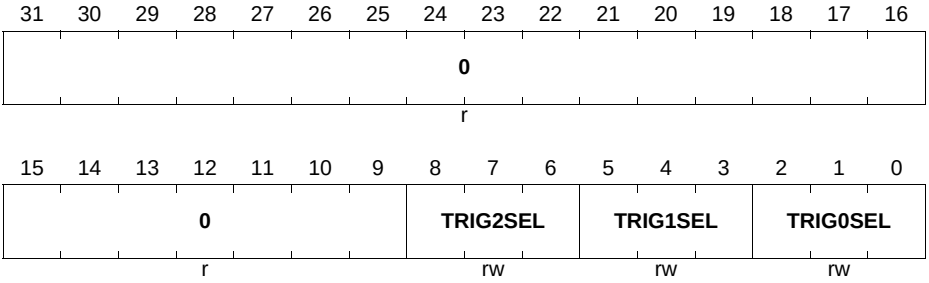
Register Short Name	Register Long Name	Offset Addr.	Access Mode		Reset	Page Num.
			Read	Write		
CCU6061 Module Registers (only available in the address range of CCU60)						
CCU60_MOSEL	CCU60 Module Output Select Register	0C$_H$	U, SV	U, SV, P	Application Reset	25-139

25.11.1.1 Module Registers

Module Output Select Register

MOSEL contains bit fields to select the output signal from module CCU6061 for the trigger signals to the A/D converters in the VADC module.

CCU60_MOSEL
CCU60 Module Output Select Register($0C_H$) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
TRIG0SEL	[2:0]	rw	**Output Trigger Select for CCU6061 TRIG0** This bit field defines the output signal from the CCU6061 module used as the trigger signal to the VADC inputs. 000_B The signal CCU60_COUT63 is selected. 001_B The signal CCU61_COUT63 is selected. 010_B The signal CCU60_CC60 is selected. 011_B The signal CCU61_CC60 is selected. 100_B The signal CCU60_SR1 is selected. 101_B The signal CCU61_SR1 is selected. 110_B The signal CCU60_SR3 is selected. 111_B The signal CCU61_SR3 is selected.
TRIG1SEL	[5:3]	rw	**Output Trigger Select for CCU6061 TRIG1** This bit field defines the output signal from the CCU6061 module used as the trigger signal to the VADC inputs. 000_B The signal CCU60_COUT63 is selected. 001_B The signal CCU61_COUT63 is selected. 010_B The signal CCU60_CC61 is selected. 011_B The signal CCU61_CC61 is selected. 100_B The signal CCU60_SR1 is selected. 101_B The signal CCU61_SR1 is selected. 110_B The signal CCU60_SR3 is selected. 111_B The signal CCU61_SR3 is selected.
TRIG2SEL	[8:6]	rw	**Output Trigger Select for CCU6061 TRIG2** This bit field defines the output signal from the CCU6061 module used as the trigger signal to the VADC inputs. 000_B The signal CCU60_COUT63 is selected. 001_B The signal CCU61_COUT63 is selected. 010_B The signal CCU60_CC62 is selected. 011_B The signal CCU61_CC62 is selected. 100_B The signal CCU60_SR1 is selected. 101_B The signal CCU61_SR1 is selected. 110_B The signal CCU60_SR3 is selected. 111_B The signal CCU61_SR3 is selected.
0	[31:9]	r	**Reserved** Returns 0 if read, should be written with 0.

*Note: CCU60_MOSEL is located in the address space of kernel CCU60. Therefore, when a reset of kernel CCU60 is triggered via registers CCU60_**KRST0** and CCU60_**KRST1**, register CCU60_MOSEL is also reset.*

*CCU60_MOSEL is **not** reset when a reset of kernel CCU61 is triggered via registers CCU61_KRST0 and CCU61_KRST1.*
*Because CCU60_MOSEL controls both signals from kernel CCU60 and CCU61, the output signals TRIG0..2 will be set to their inactive levels during **any** kernel reset of CCU60 as well as of CCU61 (see also **Figure 25-46**).*
Depending on the application, it may make sense to always reset both kernels in this case.

25.11.2 Module Output Select

For CCU6061 module, there are 3 trigger signals which are selectable from the output signals of the CCU60 and CCU61 kernels.

Each of the trigger signals (TRIG0, TRIG1, TRIG2) of the module is routed to each converter of the VADC module, as follows:

- TRIG0 of CCU6061 is routed to inputs BGREQGTC and GxREQGTC, x = 0, 1
- TRIG1 of CCU6061 is routed to inputs BGREQGTD and GxREQGTD, x = 0, 1
- TRIG2 of CCU6061 is routed to inputs BGREQGTE and GxREQGTE, x = 0, 1

Note: For ADAS devices, x = 0..3

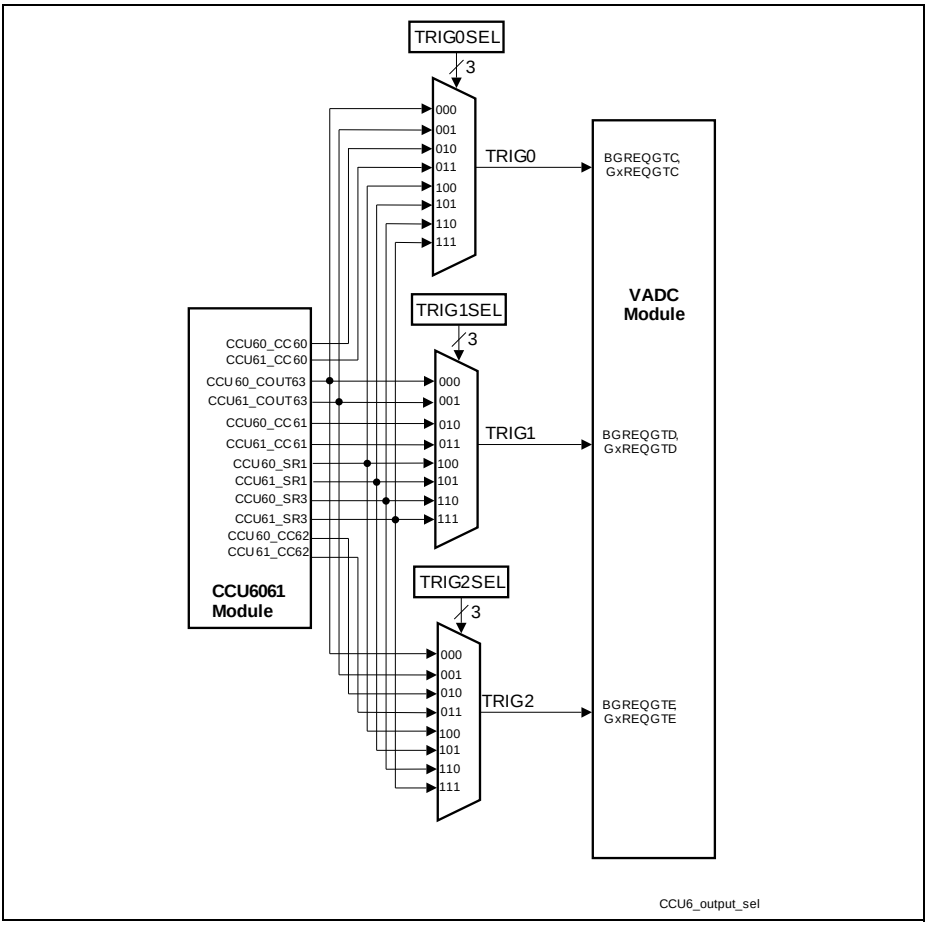


Figure 25-46 Output select trigger TC21x/TC22x/TC23x

25.11.3 Synchronous Start

Synchronous start of the capture/compare timers is supported by control bit SYSCON.CCTRIG0 in the SCU module. Bit SYSCON.CCTRIG0 is connected to the T12HR and T13HR inputs of the CCU60 and CCU61 kernels.

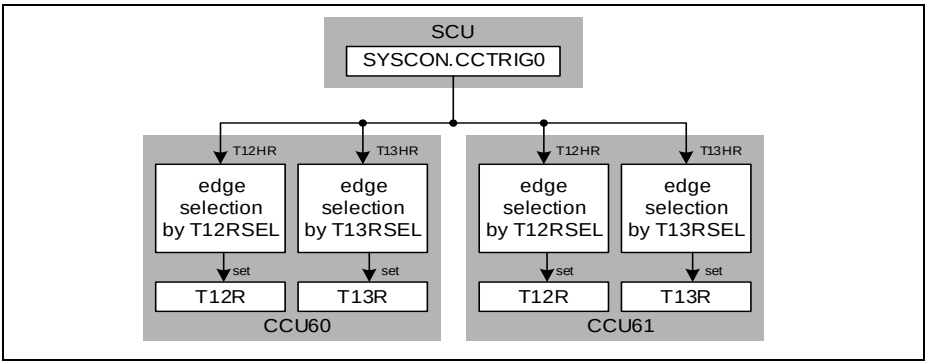


Figure 25-47 Synchronization Concept in TC21x/TC22x/TC23x

25.11.4 Digital Connections

The following tables show the digital connections of the CCU6x module with other modules or pins in the TC21x/TC22x/TC23x device.
Each input signal can be selected among 4 or 8 possible input lines, e.g. the input vector for input signal CC60IN is composed of CC60IN[D:A], whereas the input vectors for T12HR and T13HR are composed of T12HR[H:A] and T13HR[H:A]. The following sections refer to the interface signals.

The CCU6x module is clocked with the SPB_Bus clock, so $f_{CC6} = f_{SPB}$.

Note: All functional inputs of the CCU6 are synchronized to f_{CC6} before they can affect the module internal logic. The resulting delay of $2/f_{CC6}$ and an uncertainty of $1/f_{CC6}$ have to be taken into account for precise timing calculation.
An edge of an input signal can only be correctly detected if both, the high phase and the low phase of the input signal are each longer than $1/f_{CC6}$.

25.11.4.1 Connections of CCU60

This table describes the module interconnections of CCU60.

Note: Signals that are not available in the 100- and 80-pin package variant are specifically marked, see footnote [1].

Note: Signals that are not available in the 80-pin package variant are specifically marked, see footnote [2].

Table 25-17 CCU60 Digital Connections in TC21x/TC22x/TC23x

Signal	from/to Module	I/O to CCU60	Can be used to/as
CC60INA	P02.0	I	input signals for capture event on channel CC60
CC60INB	P00.1[1)]	I	
CC60INC	P02.6	I	
CC60IND	evr_wut_trigger_o	I	WUT counter underflow event trigger; may be used e.g. for trimming purposes
CC61INA	P02.2	I	input signals for capture event on channel CC61
CC61INB	P00.3[1)]	I	
CC61INC	P02.7	I	
CC61IND	0	I	

Table 25-17 CCU60 Digital Connections in TC21x/TC22x/TC23x (cont'd)

Signal	from/to Module	I/O to CCU60	Can be used to/as
CC62INA	P02.4	I	input signals for capture event on channel CC62
CC62INB	P00.5[1)]	I	
CC62INC	P02.8	I	
CC62IND	ERU_PDOUT4	I	
CTRAPA	P13.0[2)]	I	input signals for CTRAP
CTRAPB	CCU60_whe_n[3)]	I	
CTRAPC	VADCG0BFL3	I	
CTRAPD	ERU_PDOUT0	I	
CCPOS0A	P02.6	I	input signals for CCPOS0
CCPOS0B	CCU61_SR2	I	
CCPOS0C	P13.1[2)]	I	
CCPOS0D	P40.11 (SENT0A)	I	
CCPOS1A	P02.7	I	input signals for CCPOS1
CCPOS1B	P41.0 (SENT1A)	I	
CCPOS1C	P13.2[2)]	I	
CCPOS1D	0	I	
CCPOS2A	P02.8	I	input signals for CCPOS2
CCPOS2B	0	I	
CCPOS2C	P13.3[2)]	I	
CCPOS2D	0	I	
T12HRA	SCU_CCTRIG0	I	input signals for T12HR
T12HRB	P00.7[1)]	I	
T12HRC	P00.9[1)]	I	
T12HRD	GTM_TOM0_8	I	
T12HRE	P00.0	I	
T12HRF	GPT120_T6OFL	I	
T12HRG	CCU61_SR2	I	
T12HRH	ERU_PDOUT0	I	

Table 25-17 CCU60 Digital Connections in TC21x/TC22x/TC23x (cont'd)

Signal	from/to Module	I/O to CCU60	Can be used to/as
T13HRA	SCU_CCTRIG0	I	input signals for T13HR
T13HRB	P00.8[1)]	I	
T13HRC	P00.9[1)]	I	
T13HRD	GTM_TOM1_8	I	
T13HRE	0	I	
T13HRF	GPT120_T6OFL	I	
T13HRG	CCU61_SR2	I	
T13HRH	CCU60_SR1	I	
CC60	P02.0 P02.6 P11.12 P15.6[1)] (see port chapter)	O	compare outputs of channel CC60
	VADC		VADC gating input, see **CCU60_MOSEL** for the respective trigger select
COUT60	P02.1 P11.9 P15.7[1)] (see port chapter)	O	
	ERU_IN01		
CC61	P02.2 P02.7 P11.11 P15.5 (see port chapter)	O	compare outputs of channel CC61
	VADC		VADC gating input, see **CCU60_MOSEL** for the respective trigger select
COUT61	P02.3 P11.6 P15.8[1)] (see port chapter)	O	

Table 25-17 CCU60 Digital Connections in TC21x/TC22x/TC23x (cont'd)

Signal	from/to Module	I/O to CCU60	Can be used to/as
CC62	P02.4 P02.8 P11.10 P15.4[1)] (see port chapter	O	compare outputs of channel CC62
	VADC		VADC gating input, see **CCU60_MOSEL** for the respective trigger select
COUT62	P02.5 P11.3 P14.0 (see port chapter	O	
COUT63	P00.0 P11.2 P14.1 (see port chapter)	O	compare output of channel CC63
	VADC		VADC gating input, see **CCU60_MOSEL** for the respective trigger select
LBE	n.c.	O	lost bit event output
whe_n[3)]	CCU60_CTRAPB	O	wrong Hall event output (inverted)
OTGB0[0:15]	OCDS	O	OCDS Trigger Bus 0
OTGB1[0:15]	OCDS	O	OCDS Trigger Bus 1
SR0	Interrupt Router: SRC_CCU60SR0	O	CCU60 Service Request 0
SR1	CCU60_T13HRH	O	T13 count trigger
	VADC		VADC gating input, see **CCU60_MOSEL** for the respective trigger select
	Interrupt Router: SRC_CCU60SR1		CCU60 Service Request 1

Table 25-17 CCU60 Digital Connections in TC21x/TC22x/TC23x (cont'd)

Signal	from/to Module	I/O to CCU60	Can be used to/as
SR2	CCU61_CCPOS0B, CCU61_T12HRG, CCU61_T13HRG	O	CCU61 triggers
	Interrupt Router: SRC_CCU60SR2		CCU60 Service Request 2
SR3	VADC_GxREQTRA (x = 0..1 (3 for ADAS devices)), VADC_BGREQTRA	O	VADC trigger capability, for additional options see also **CCU60_MOSEL**
	Interrupt Router: SRC_CCU60SR3		CCU60 Service Request 3

1) not available for devices in 100-pin and 80-pin package

2) not available for devices in 80-pin package

3) CCU6x_whe_n is the inverted version of the internal Wrong Hall Event signal whe (to match the polarity requirements of the CTRAP input)

25.11.4.2 Connections of CCU61

This table describes the module interconnections of CCU61.

Note: Signals that are not available in the 100- and 80-pin package variant are specifically marked, see footnote [1].

Note: Signals that are not available in the 80-pin package variant are specifically marked, see footnote [3].

Table 25-18 CCU61 Digital Connections in TC21x/TC22x/TC23x

Signal	from/to Module	I/O to CCU61	Can be used to/as
CC60INA	P00.1[1]	I	input signals for capture event on channel CC60
CC60INB	P02.0	I	
CC60INC	P00.7[1]	I	
CC60IND	evr_wut_trigger_o	I	WUT counter underflow event trigger; may be used e.g. for trimming purposes
CC61INA	P00.3[1]	I	input signals for capture event on channel CC61
CC61INB	P02.2	I	
CC61INC	P00.8[1]	I	
CC61IND	0	I	
CC62INA	P00.5[1]	I	input signals for capture event on channel CC62
CC62INB	P02.4	I	
CC62INC	P00.9[1]	I	
CC62IND	ERU_PDOUT5	I	
CTRAPA	P00.0	I	input signals for CTRAP
CTRAPB	CCU61_whe_n[2]	I	
CTRAPC	P33.4[1]	I	
CTRAPD	ERU_PDOUT1	I	
CCPOS0A	P00.7[1]	I	input signals for CCPOS0
CCPOS0B	CCU60_SR2	I	
CCPOS0C	P33.7	I	
CCPOS0D	0	I	

Table 25-18 CCU61 Digital Connections in TC21x/TC22x/TC23x (cont'd)

Signal	from/to Module	I/O to CCU61	Can be used to/as
CCPOS1A	P00.8[1)]	I	input signals for CCPOS1
CCPOS1B	P41.2 (SENT2A)	I	
CCPOS1C	P33.6	I	
CCPOS1D	P41.3 (SENT3A)	I	
CCPOS2A	P00.9[1)]	I	input signals for CCPOS2
CCPOS2B	0	I	
CCPOS2C	P33.5	I	
CCPOS2D	0	I	
T12HRA	SCU_CCTRIG0	I	input signals for T12HR
T12HRB	P02.6	I	
T12HRC	P02.8	I	
T12HRD	0	I	
T12HRE	0	I	
T12HRF	GPT120_T6OFL	I	
T12HRG	CCU60_SR2	I	
T12HRH	ERU_PDOUT1	I	
T13HRA	SCU_CCTRIG0	I	input signals for T13HR
T13HRB	P02.7	I	
T13HRC	P02.8	I	
T13HRD	GTM_TOM0_9	I	
T13HRE	0	I	
T13HRF	GPT120_T6OFL	I	
T13HRG	CCU60_SR2	I	
T13HRH	CCU61_SR1	I	

Table 25-18 CCU61 Digital Connections in TC21x/TC22x/TC23x (cont'd)

Signal	from/to Module	I/O to CCU61	Can be used to/as
CC60	P00.1[1)] P00.7[1)] P11.12 P20.8 P33.3[1)] P33.5 (see port chapter)	O	compare outputs of channel CC60
	VADC		VADC gating input, see **CCU60_MOSEL** for the respective trigger select
COUT60	P00.2[1)] P11.9 P20.11 P33.7 P33.12[1)] (see port chapter)	O	
	ERU_IN11		
CC61	P00.3[1)] P00.8[1)] P11.11 P20.9 P33.6 P33.11[1)] (see port chapter)	O	compare outputs of channel CC61
	VADC		VADC gating input, see **CCU60_MOSEL** for the respective trigger select
COUT61	P00.4[1)] P11.6 P20.12 P33.10 (see port chapter)	O	

Table 25-18 CCU61 Digital Connections in TC21x/TC22x/TC23x (cont'd)

Signal	from/to Module	I/O to CCU61	Can be used to/as
CC62	P00.5[1)] P00.9[1)] P11.10 P20.10[3)] P33.9 (see port chapter)	O	compare outputs of channel CC62
	VADC		VADC gating input, see **CCU60_MOSEL** for the respective trigger select
COUT62	P00.6[1)] P11.3 P20.13 P33.8 (see port chapter)	O	
COUT63	P00.12[1)] P11.2 P20.7[1)] P33.2[1)] (see port chapter)	O	compare output of channel CC63
	VADC		VADC gating input, see **CCU60_MOSEL** for the respective trigger select
LBE	n.c.	O	lost bit event output
whe_n[2)]	CCU61_CTRAPB	O	wrong Hall event output (inverted)
OTGB0[0:15]	OCDS	O	OCDS Trigger Bus 0
OTGB1[0:15]	OCDS	O	OCDS Trigger Bus 1
SR0	Interrupt Router: SRC_CCU61SR0	O	CCU61 Service Request 0
SR1	CCU61_T13HRH	O	T13 count trigger
	VADC		VADC gating input, see **CCU60_MOSEL** for the respective trigger select
	Interrupt Router: SRC_CCU61SR1		CCU61 Service Request 1

Table 25-18 CCU61 Digital Connections in TC21x/TC22x/TC23x (cont'd)

Signal	from/to Module	I/O to CCU61	Can be used to/as
SR2	CCU60_CCPOS0B, CCU60_T12HRG, CCU60_T13HRG	O	CCU60 triggers
	Interrupt Router: SRC_CCU61SR2		CCU61 Service Request 2
SR3	VADC_GxREQTRB (x = 0..1 (3 for ADAS devices)), VADC_BGREQTRB	O	VADC trigger capability, for additional options see also **CCU60_MOSEL**
	Interrupt Router: SRC_CCU61SR3		CCU61 Service Request 3

1) not available for devices in 100-pin or 80-pin package

2) CCU6x_whe_n is the inverted version of the internal Wrong Hall Event signal whe (to match the polarity requirements of the CTRAP input)

3) not available for devices in 80-pin package

26 General Purpose Timer Unit (GPT12)

The General Purpose Timer Unit blocks GPT1 and GPT2 have very flexible multifunctional timer structures which may be used for timing, event counting, pulse width measurement, pulse generation, frequency multiplication, and other purposes.

They incorporate five 16-bit timers that are grouped into the two timer blocks GPT1 and GPT2. Each timer in each block may operate independently in a number of different modes such as Gated Timer or Counter Mode, or may be concatenated with another timer of the same block.

Each block has alternate input/output functions and specific interrupts (service requests) associated with it. Input signals can be selected from several sources by register PISEL.

The GPT12 module is clocked with clock f_{GPT}.

Block GPT1 contains three timers/counters: The core timer T3 and the two auxiliary timers T2 and T4. The maximum resolution is $f_{GPT}/4$. The auxiliary timers of GPT1 may optionally be configured as reload or capture registers for the core timer. These registers are listed in **Section 26.1.6.1**.

The following list summarizes the supported features:

- $f_{GPT}/4$ maximum resolution
- 3 independent timers/counters
- Timers/counters can be concatenated
- 4 operating modes:
  - Timer Mode
  - Gated Timer Mode
  - Counter Mode
  - Incremental Interface Mode
- Reload and Capture functionality
- Separate interrupts

Block GPT2 contains two timers/counters: The core timer T6 and the auxiliary timer T5. The maximum resolution is $f_{GPT}/2$. An additional Capture/Reload register (CAPREL) supports capture and reload operation with extended functionality. These registers are listed in **Section 26.2.7.1**.

The following list summarizes the supported features:

- $f_{GPT}/2$ maximum resolution
- 2 independent timers/counters
- Timers/counters can be concatenated
- 3 operating modes:
  - Timer Mode
  - Gated Timer Mode
  - Counter Mode
- Extended capture/reload functions via 16-bit capture/reload register CAPREL
- Separate interrupts

26.1 Timer Block GPT1

All three timers of block GPT1 (T2, T3, T4) can run in one of 4 basic modes: Timer Mode, Gated Timer Mode, Counter Mode, or Incremental Interface Mode. All timers can count up or down. Each timer of GPT1 is controlled by a separate control register TxCON.

Each timer has an input pin TxIN (alternate pin function) associated with it, which serves as the gate control in Gated Timer Mode, or as the count input in Counter Mode. The count direction (up/down) may be programmed via software or may be dynamically altered by a signal at the External Up/Down control input TxEUD (alternate pin function). An overflow/underflow of core timer T3 is indicated by the Output Toggle Latch T3OTL, whose state may be output on the associated pin T3OUT (alternate pin function). The auxiliary timers T2 and T4 may additionally be concatenated with the core timer T3 (through T3OTL) or may be used as capture or reload registers for the core timer T3.

The current contents of each timer can be read or modified by the CPU by accessing the corresponding timer count registers T2, T3, or T4. When any of the timer registers is written to by the CPU in the state immediately preceding a timer increment, decrement, reload, or capture operation, the CPU write operation has priority in order to guarantee correct results.

The interrupt requests of GPT1 are signalled on service request lines SR0, SR1, and SR2.

The input and output lines of GPT1 are connected to pins. The control registers for the port functions are located in the respective port modules.

*Note: The timing requirements for external input signals can be found in **Section 26.1.5**, **Section 26.5.2** summarizes the module interface signals, including pins and interrupt request signals.*

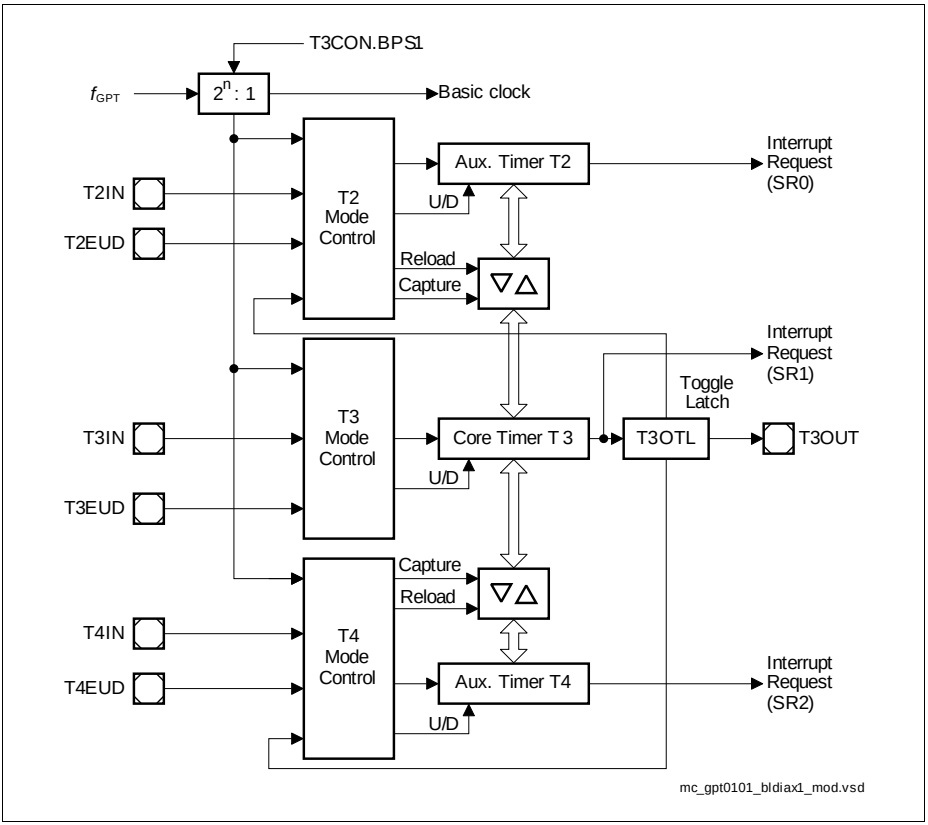


Figure 26-1 GPT1 Block Diagram

26.1.1 GPT1 Core Timer T3 Control

The current contents of the core timer T3 are reflected by its count register T3. This register can also be written to by the CPU, for example, to set the initial start value.

The core timer T3 is configured and controlled via its control register T3CON.

Timer T3 Run Control

The core timer T3 can be started or stopped by software through bit T3R (Timer T3 Run Bit). This bit is relevant in all operating modes of T3. Setting bit T3R will start the timer, clearing bit T3R stops the timer.

In Gated Timer Mode, the timer will only run if T3R = 1 and the gate is active (high or low, as programmed).

Note: When bit T2RC or T4RC in timer control register T2CON or T4CON is set, bit T3R will also control (start and stop) the auxiliary timer(s) T2 and/or T4.

Count Direction Control

The count direction of the GPT1 timers (core timer and auxiliary timers) can be controlled either by software or by the external input pin TxEUD (Timer Tx External Up/Down Control Input). These options are selected by bits TxUD and TxUDE in the respective control register TxCON. When the up/down control is provided by software (bit TxUDE = 0), the count direction can be altered by setting or clearing bit TxUD. When bit TxUDE = 1, pin TxEUD is selected to be the controlling source of the count direction. However, bit TxUD can still be used to reverse the actual count direction, as shown in **Table 26-6**. The count direction can be changed regardless of whether or not the timer is running.

Note: When pin TxEUD is used as external count direction control input, it must be configured as input.

Timer T3 Output Toggle Latch

The overflow/underflow signal of timer T3 is connected to a block named ‘Toggle Latch’, shown in the Timer Mode diagrams. **Figure 26-2** illustrates the details of this block. An overflow or underflow of T3 will clock two latches: The first latch represents bit T3OTL in control register T3CON. The second latch is an internal latch toggled by T3OTL’s output. Both latch outputs are connected to the input control blocks of the auxiliary timers T2 and T4. The output level of the shadow latch will match the output level of T3OTL, but is delayed by one clock cycle. When the T3OTL value changes, this will result in a temporarily different output level from T3OTL and the shadow latch, which can trigger the selected count event in T2 and/or T4.

When software writes to T3OTL, both latches are set or cleared simultaneously. In this case, both signals to the auxiliary timers carry the same level and no edge will be detected. Bit T3OE (overflow/underflow output enable) in register T3CON enables the state of T3OTL to be monitored via an external pin T3OUT. When T3OTL is linked to an external port pin (must be configured as output), T3OUT can be used to control external HW. If T3OE = 1, pin T3OUT outputs the state of T3OTL. If T3OE = 0, pin T3OUT outputs a high level (as long as the T3OUT alternate function is selected for the port pin).

The trigger signals can serve as an input for the counter function or as a trigger source for the reload function of the auxiliary timers T2 and T4.

As can be seen from **Figure 26-2**, when latch T3OTL is modified by software to determine the state of the output line, also the internal shadow latch is set or cleared accordingly. Therefore, no trigger condition is detected by T2/T4 in this case.

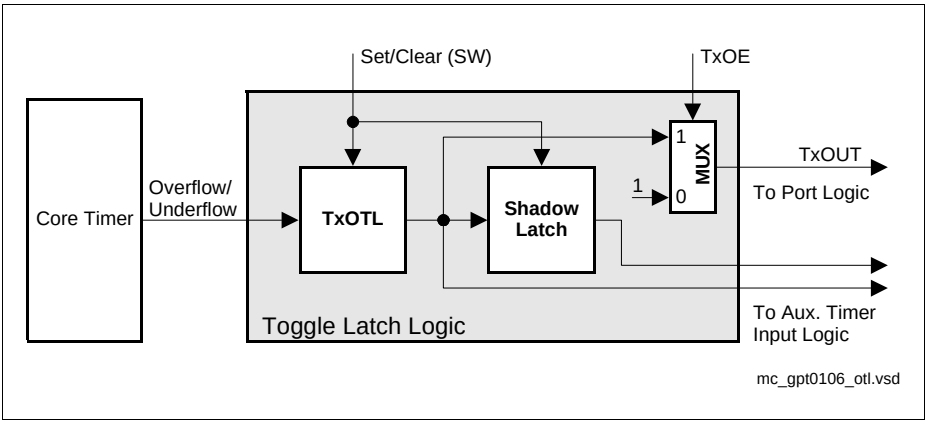


Figure 26-2 Block Diagram of the Toggle Latch Logic of Core Timer T3 (x = 3)

26.1.2 GPT1 Core Timer T3 Operating Modes

Timer T3 can operate in one of several modes.

Timer T3 in Timer Mode

Timer Mode for the core timer T3 is selected by setting bitfield T3M in register T3CON to 000_B. In Timer Mode, T3 is clocked with the module's input clock f_{GPT} divided by two programmable prescalers controlled by bitfields BPS1 and T3I in register T3CON. Please see **Section 26.1.5** for details on the input clock options.

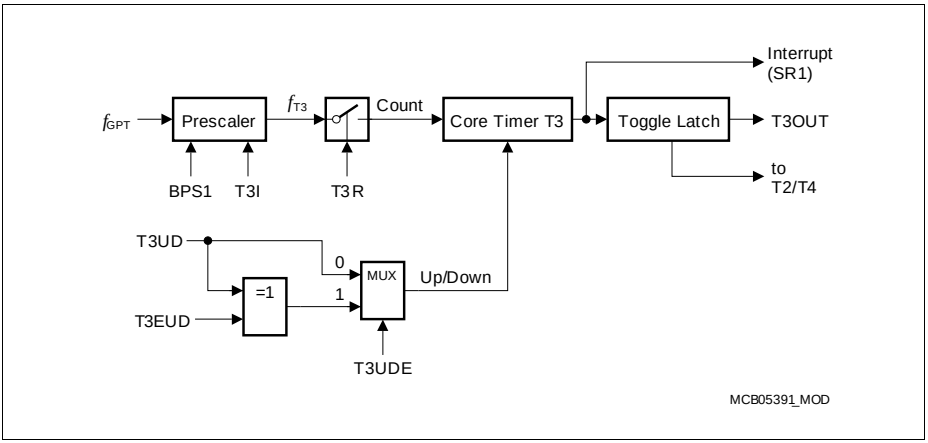


Figure 26-3 Block Diagram of Core Timer T3 in Timer Mode

Timer T3 in Gated Timer Mode

Gated Timer Mode for the core timer T3 is selected by setting bitfield T3M in register T3CON to 010_B or 011_B. Bit T3M.0 (T3CON.3) selects the active level of the gate input. The same options for the input frequency are available in Gated Timer Mode as in Timer Mode (see **Section 26.1.5**). However, the input clock to the timer in this mode is gated by the external input pin T3IN (Timer T3 External Input).

To enable this operation, the associated pin T3IN must be configured as input.

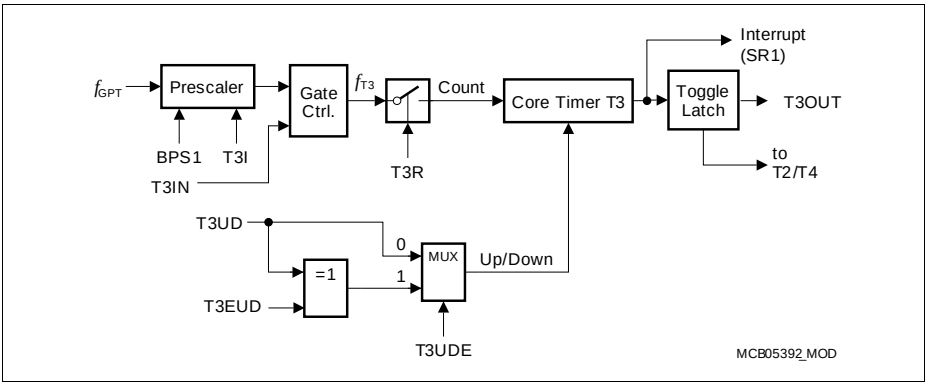


Figure 26-4 Block Diagram of Core Timer T3 in Gated Timer Mode

If T3M = 010_B, the timer is enabled when T3IN shows a low level. A high level at this line stops the timer. If T3M = 011_B, line T3IN must have a high level in order to enable the timer. Additionally, the timer can be turned on or off by software using bit T3R. The timer will only run if T3R is 1 and the gate is active. It will stop if either T3R is 0 or the gate is inactive.

Note: A transition of the gate signal at pin T3IN does not cause a service request via SR1.

Timer T3 in Counter Mode

Counter Mode for the core timer T3 is selected by setting bitfield T3M in register T3CON to 001_B. In Counter Mode, timer T3 is clocked by a transition at the external input pin T3IN. The event causing an increment or decrement of the timer can be a positive, a negative, or both a positive and a negative transition at this line. Bitfield T3I in control register T3CON selects the triggering transition (see **Table 26-8**).

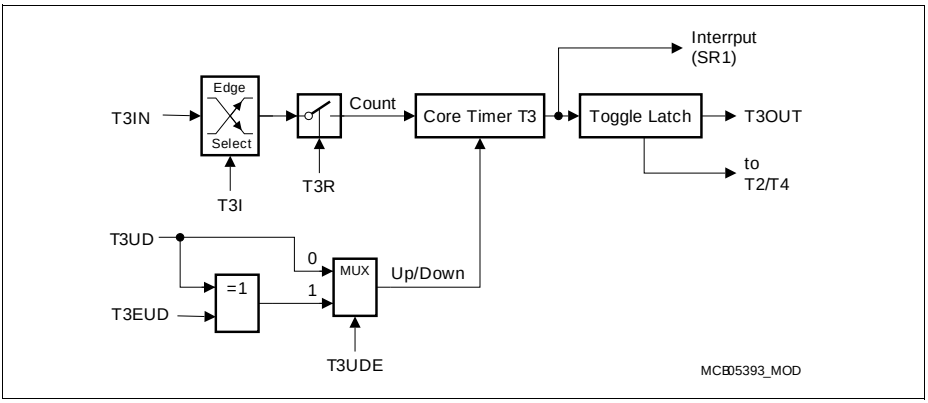


Figure 26-5 Block Diagram of Core Timer T3 in Counter Mode

For Counter Mode operation, pin T3IN must be configured as input. The maximum input frequency allowed in Counter Mode depends on the selected prescaler value. To ensure that a transition of the count input signal applied to T3IN is recognized correctly, its level must be held high or low for a minimum number of module clock cycles before it changes. This information can be found in **Section 26.1.5**.

Timer T3 in Incremental Interface Mode

Incremental Interface Mode for the core timer T3 is selected by setting bitfield T3M in register T3CON to 110_B or 111_B. In Incremental Interface Mode, the two inputs associated with core timer T3 (T3IN, T3EUD) are used to interface to an incremental encoder. T3 is clocked by each transition on one or both of the external input pins to provide 2-fold or 4-fold resolution of the encoder input.

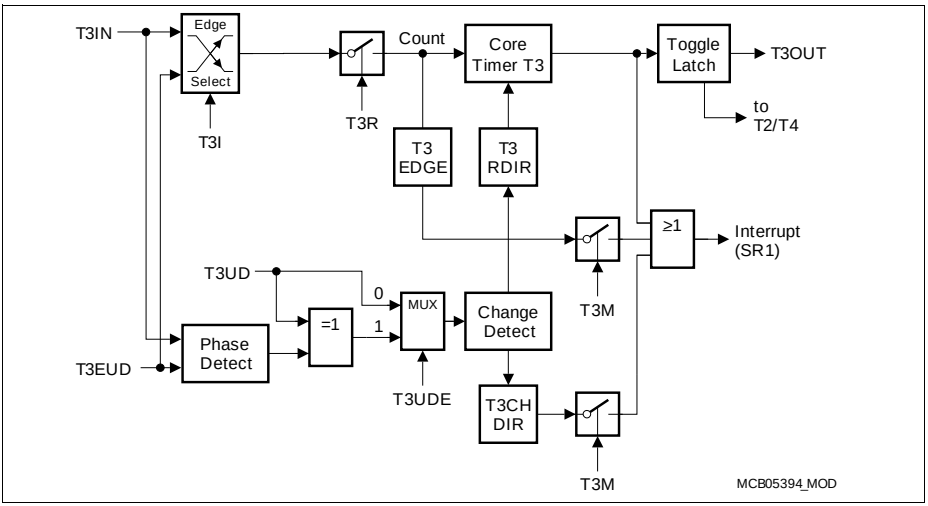


Figure 26-6 Block Diagram of Core Timer T3 in Incremental Interface Mode

Bitfield T3I in control register T3CON selects the triggering transitions (see **Table 26-10**). The sequence of the transitions of the two input signals is evaluated and generates count pulses as well as the direction signal. So T3 is modified automatically according to the speed and the direction of the incremental encoder and, therefore, its contents always represent the encoder's current position.

The service request generation can be selected: In Rotation Detection Mode (T3M = 110_B), a service request is generated each time the count direction of T3 changes. In Edge Detection Mode (T3M = 111_B), a service request is generated each time a count edge for T3 is detected. Count direction, changes in the count direction, and count requests are monitored by status bits T3RDIR, T3CHDIR, and T3EDGE in register T3CON.

The incremental encoder can be connected directly to the TC21x/TC22x/TC23x without external interface logic. In a standard system, however, comparators will be employed to convert the encoder's differential outputs (such as A, $\overline{A}$) to digital signals (such as A). This greatly increases noise immunity.

Note: The third encoder output T0, that indicates the mechanical zero position, may be connected to an external interrupt input and trigger a reset of timer T3.
*If input T4IN is available, T0 can be connected there and clear T3 automatically without requiring an interrupt (see bit CLRT3EN in register **T4CON**).*

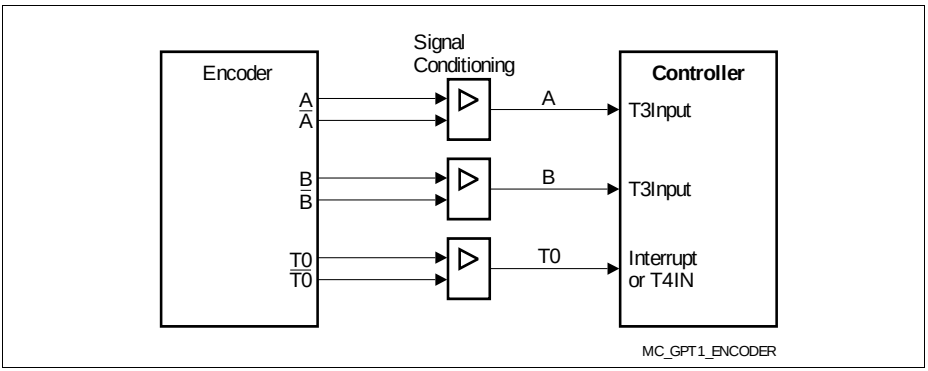


Figure 26-7 Connection of the Encoder to the TC21x/TC22x/TC23x

For Incremental Interface Mode operation, the following conditions must be met:

- Bitfield T3M must be 110_B or 111_B.
- Both pins T3IN and T3EUD must be configured as input.
- Pin T4IN must be configured as input, if used for T0.
- Bit T3UDE must be 1 to enable automatic external direction control.

The maximum count frequency allowed in Incremental Interface Mode depends on the selected prescaler value. To ensure that a transition of any input signal is recognized correctly, its level must be held high or low for a minimum number of module clock cycles before it changes. This information can be found in **Section 26.1.5**.

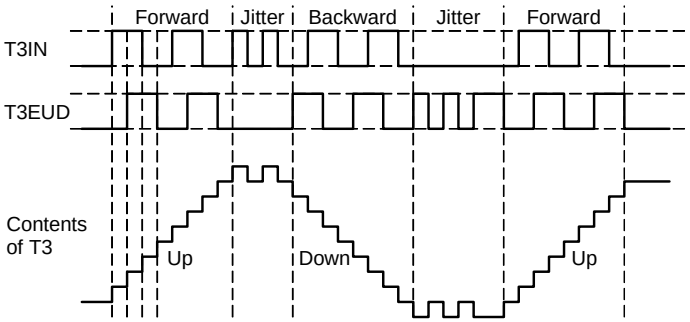
As in Incremental Interface Mode two input signals with a 90° phase shift are evaluated, their maximum input frequency can be half the maximum count frequency.

In Incremental Interface Mode, the count direction is automatically derived from the sequence in which the input signals change, which corresponds to the rotation direction of the connected sensor. **Table 26-1** summarizes the possible combinations.

Table 26-1 GPT1 Core Timer T3 (Incremental Interface Mode) Count Direction

Level on Respective other Input	T3IN Input		T3EUD Input	
	Rising ↑	**Falling ↓**	**Rising ↑**	**Falling ↓**
High	Down	Up	Up	Down
Low	Up	Down	Down	Up

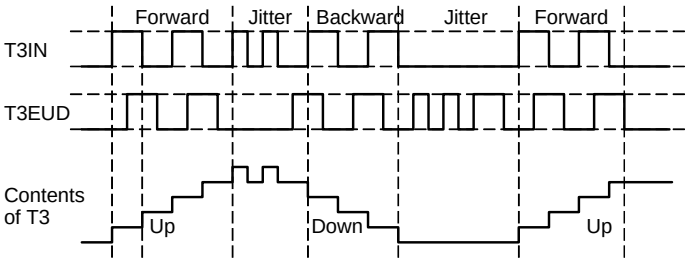
Figure 26-8 and **Figure 26-9** give examples of T3's operation, visualizing count signal generation and direction control. They also show how input jitter is compensated, which might occur if the sensor rests near to one of its switching points.



Note: This example shows the timer behaviour assuming that T3 counts upon any transition on any input (T3IN and T3EUD), i.e. T3I = 011_B.

MCT04373

Figure 26-8 Evaluation of Incremental Encoder Signals, 2 Count Inputs



Note: This example shows the timer behaviour assuming that T3 counts upon any transition on input T3IN, i.e. T3I = 001_B.

MCT04374

Figure 26-9 Evaluation of Incremental Encoder Signals, 1 Count Input

Note: Timer T3 operating in Incremental Interface Mode automatically provides information on the sensor's current position. Dynamic information (speed,

acceleration, deceleration) may be obtained by measuring the incoming signal periods (see ***"Combined Capture Modes" on Page 26-58****).*

26.1.3 GPT1 Auxiliary Timers T2/T4 Control

Auxiliary timers T2 and T4 have exactly the same functionality. They can be configured for Timer Mode, Gated Timer Mode, Counter Mode, or Incremental Interface Mode with the same options for the timer frequencies and the count signal as the core timer T3. In addition to these 4 counting modes, the auxiliary timers can be concatenated with the core timer, or they may be used as reload or capture registers in conjunction with the core timer. The start/stop function of the auxiliary timers can be remotely controlled by the T3 run control bit. Several timers may thus be controlled synchronously.

The current contents of an auxiliary timer are reflected by its count register T2 or T4, respectively. These registers can also be written to by the CPU, for example, to set the initial start value.

The individual configurations for timers T2 and T4 are determined by their control registers T2CON and T4CON, that are organized identically. Note that functions which are present in all 3 timers of block GPT1 are controlled in the same bit positions and in the same manner in each of the specific control registers.

Note: The auxiliary timers have no output toggle latch and no alternate output function.

Timer T2/T4 Run Control

Each of the auxiliary timers T2 and T4 can be started or stopped by software in two different ways:

- Through the associated timer run bit (T2R or T4R). In this case it is required that the respective control bit TxRC = 0.
- Through the core timer's run bit (T3R). In this case the respective remote control bit must be set (TxRC = 1).

The selected run bit is relevant in all operating modes of T2/T4. Setting the bit will start the timer, clearing the bit stops the timer.

In Gated Timer Mode, the timer will only run if the selected run bit is set and the gate is active (high or low, as programmed).

Note: If remote control is selected T3R will start/stop timer T3 and the selected auxiliary timer(s) synchronously.

Count Direction Control

The count direction of the GPT1 timers (core timer and auxiliary timers) is controlled in the same way, either by software or by the external input pin TxEUD. Please refer to the description in **Table 26-6**.

Note: When pin TxEUD is used as external count direction control input, it must be configured as input.

26.1.4 GPT1 Auxiliary Timers T2/T4 Operating Modes

The operation of the auxiliary timers in the basic operating modes is almost identical with the core timer's operation, with very few exceptions. Additionally, some combined operating modes can be selected.

Timers T2 and T4 in Timer Mode

Timer Mode for an auxiliary timer Tx is selected by setting its bitfield TxM in register TxCON to 000_B.

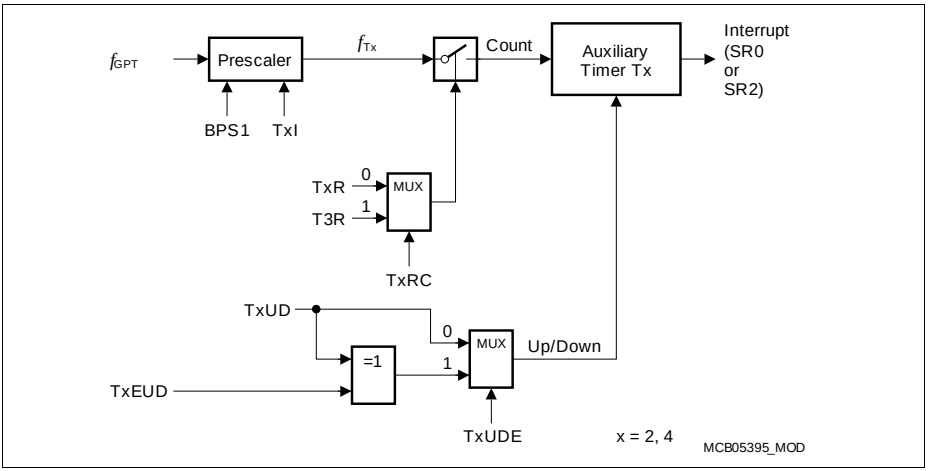


Figure 26-10 Block Diagram of an Auxiliary Timer in Timer Mode

Timers T2 and T4 in Gated Timer Mode

Gated Timer Mode for an auxiliary timer Tx is selected by setting bitfield TxM in register TxCON to 010_B or 011_B. Bit TxM.0 (TxCON.3) selects the active level of the gate input.

Note: A transition of the gate signal at TxIN does not cause a service request. Service requests of timer T2 are handled via SR0, and service requests of timer T4 are handled via SR2.

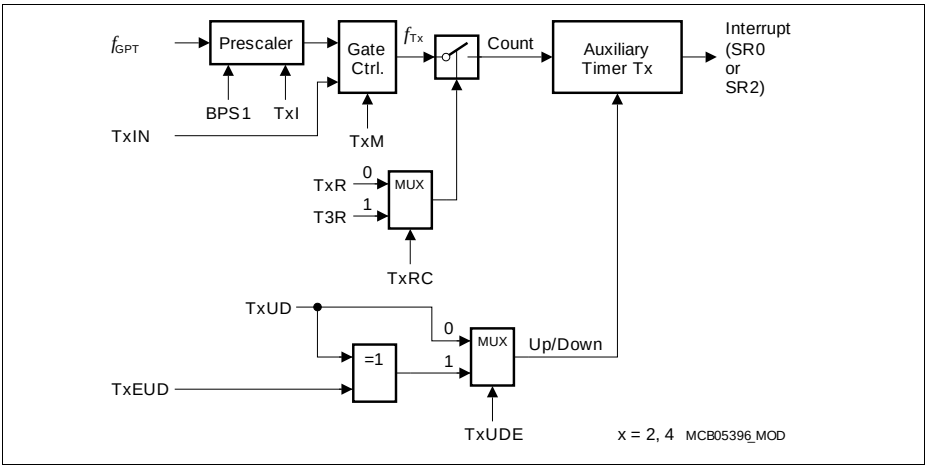


Figure 26-11 Block Diagram of an Auxiliary Timer in Gated Timer Mode

Note: There is no output toggle latch for T2 and T4.
Start/stop of an auxiliary timer can be controlled locally or remotely.

Timers T2 and T4 in Counter Mode

Counter Mode for an auxiliary timer Tx is selected by setting bitfield TxM in register TxCON to 001_B. In Counter Mode, an auxiliary timer can be clocked either by a transition at its external input line TxIN, or by a transition of timer T3's toggle latch T3OTL. The event causing an increment or decrement of a timer can be a positive, a negative, or both a positive and a negative transition at either the respective input pin or at the toggle latch. Bitfield TxI in control register TxCON selects the triggering transition (see **Table 26-9**).

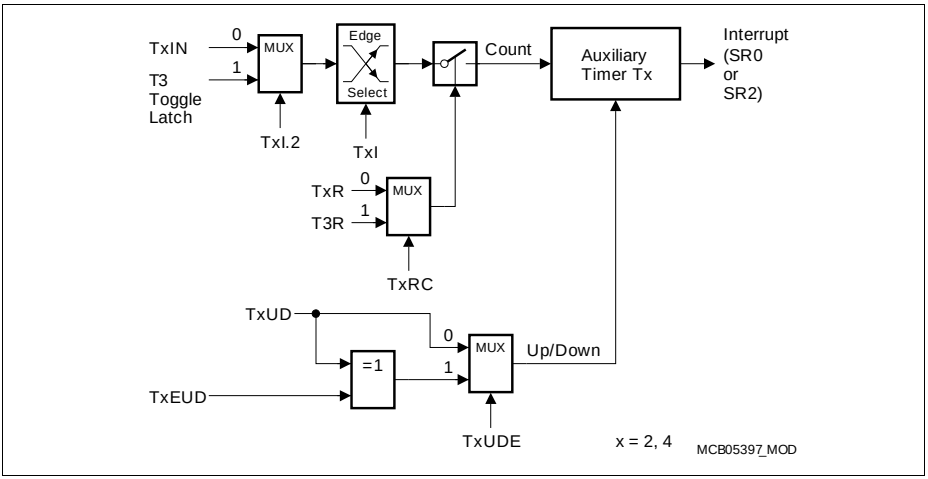


Figure 26-12 Block Diagram of an Auxiliary Timer in Counter Mode

Note: Only state transitions of T3OTL which are caused by the overflows/underflows of T3 will trigger the counter function of T2/T4. Modifications of T3OTL via software will NOT trigger the counter function of T2/T4.

For counter operation, pin TxIN must be configured as input. The maximum input frequency allowed in Counter Mode depends on the selected prescaler value. To ensure that a transition of the count input signal applied to TxIN is recognized correctly, its level must be held high or low for a minimum number of module clock cycles before it changes. This information can be found in **Section 26.1.5**.

Timer Concatenation

Using the toggle bit T3OTL as a clock source for an auxiliary timer in Counter Mode concatenates the core timer T3 with the respective auxiliary timer. This concatenation forms either a 32-bit or a 33-bit timer/counter, depending on which transition of T3OTL is selected to clock the auxiliary timer.

- **32-bit Timer/Counter:** If both a positive and a negative transition of T3OTL are used to clock the auxiliary timer, this timer is clocked on every overflow/underflow of the core timer T3. Thus, the two timers form a 32-bit timer.
- **33-bit Timer/Counter:** If either a positive or a negative transition of T3OTL is selected to clock the auxiliary timer, this timer is clocked on every second overflow/underflow of the core timer T3. This configuration forms a 33-bit timer (16-bit core timer + T3OTL + 16-bit auxiliary timer).
  As long as bit T3OTL is not modified by software, it represents the state of the internal toggle latch, and can be regarded as part of the 33-bit timer.

The count directions of the two concatenated timers are not required to be the same. This offers a wide variety of different configurations.

T3, which represents the low-order part of the concatenated timer, can operate in Timer Mode, Gated Timer Mode or Counter Mode in this case.

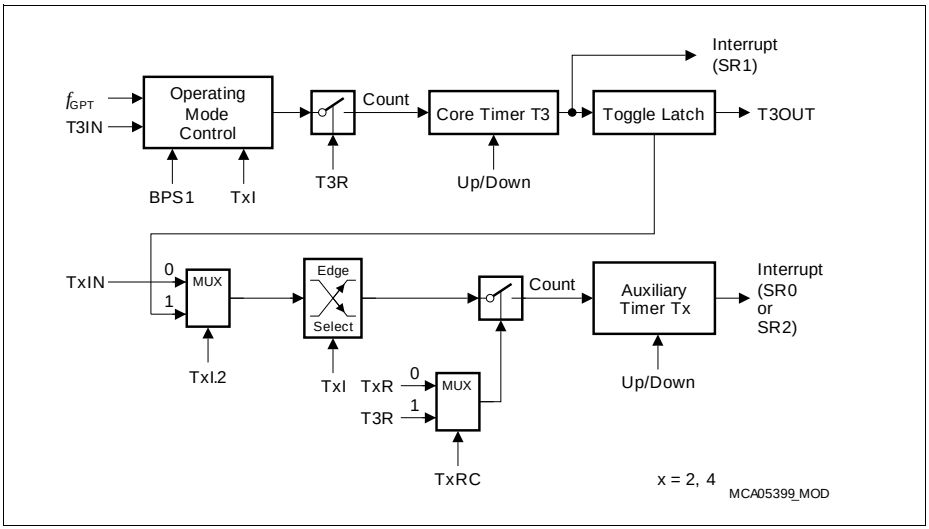


Figure 26-13 Concatenation of Core Timer T3 and an Auxiliary Timer

When reading the low and high parts of the concatenated timer, care must be taken to obtain consistent values in particular after a timer overflow/underflow (e.g. one part may already have considered an overflow, while the other has not).

Note: This is a general issue when reading multi-word results with consecutive instructions, and not necessarily unique to the GPT12 module architecture.

The following algorithm may be used to read concatenated GPT1 timers, represented by TIMER_HIGH (for auxiliary timer, here T2) and TIMER_LOW (for core timer T3). The high part is read twice, and reading of the low part is repeated if two different values were read for the high part:

- TIMER_HIGH_TMP = T2
- TIMER_LOW = T3
- Wait two basic clock cycles (to allow increment/decrement of auxiliary timer in case of core timer overflow/underflow) - see **Table 26-2**
- TIMER_HIGH = T2
- If TIMER_HIGH is not equal to TIMER_HIGH_TMP then TIMER_LOW = T3

After execution of this algorithm, TIMER_HIGH and TIMER_LOW represent a consistent time stamp of the concatenated timers.

The equivalent number of module clock cycles corresponding to two basic clock cycles is shown in **Table 26-2**.

Table 26-2 Number of Module Clock Cycles to Wait for Two Basic Clock Cycles

Block Prescaler	BPS1 = 01_B	BPS1 = 00_B	BPS1 = 11_B	BPS1 = 10_B
Number of Module clocks	8	16	32	64

In case the required timer resolution can be achieved with different combinations of the Block Prescaler BPS1 and the Individual Prescalers TxI (see **Table 26-7**), the variant with the smallest value for the Block Prescaler may be chosen to minimize the waiting time.

Timers T2 and T4 in Incremental Interface Mode

Incremental Interface Mode for an auxiliary timer Tx is selected by setting bitfield TxM in the respective register TxCON to 110_B or 111_B. In Incremental Interface Mode, the two inputs associated with an auxiliary timer Tx (TxIN, TxEUD) are used to interface to an incremental encoder. Tx is clocked by each transition on one or both of the external input pins to provide 2-fold or 4-fold resolution of the encoder input.

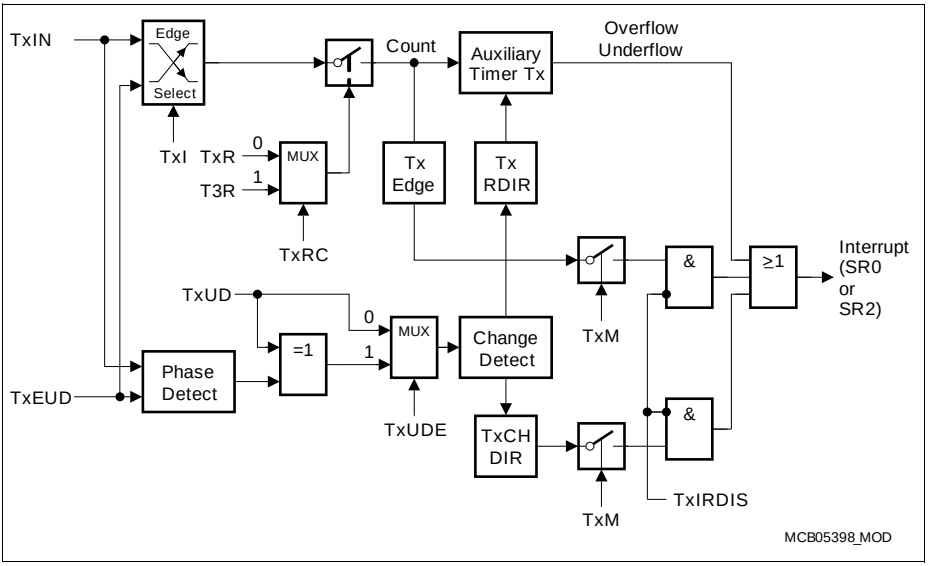


Figure 26-14 Block Diagram of an Auxiliary Timer in Incremental Interface Mode

The operation of the auxiliary timers T2 and T4 in Incremental Interface Mode and the interrupt generation are the same as described for the core timer T3. The descriptions, figures and tables apply accordingly.

Note: Timers T2 and T4 operating in Incremental Interface Mode automatically provide information on the sensor's current position. For dynamic information (speed, acceleration, deceleration) see ***"Combined Capture Modes" on Page 26-58****).*

Timers T2 and T4 in Reload Mode

Reload Mode for an auxiliary timer Tx is selected by setting bitfield TxM in the respective register TxCON to 100_B. In Reload Mode, the core timer T3 is reloaded with the contents of an auxiliary timer register, triggered by one of two different signals. The trigger signal is selected the same way as the clock source for Counter Mode (see **Table 26-9**), i.e. a transition of the auxiliary timer's input TxIN or the toggle latch T3OTL may trigger the reload.

Note: When programmed for Reload Mode, the respective auxiliary timer (T2 or T4) stops independently of its run flag T2R or T4R.
The timer input pin TxIN must be configured as input if it shall trigger a reload operation.

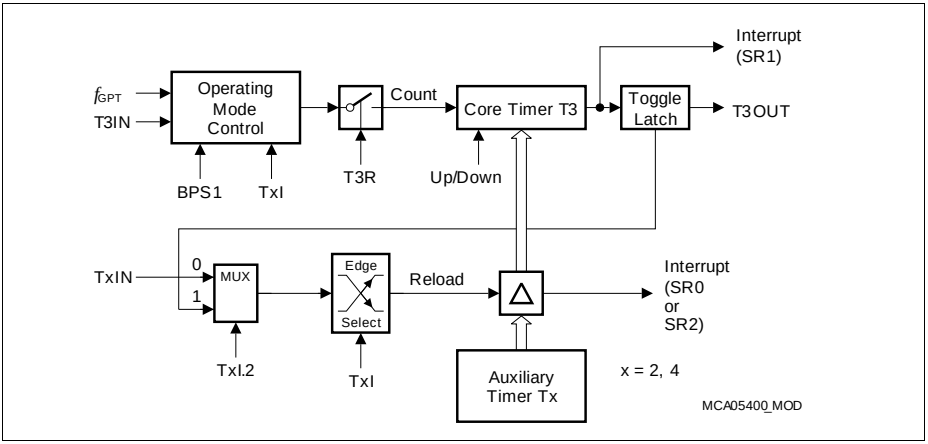


Figure 26-15 GPT1 Auxiliary Timer in Reload Mode

Upon a trigger signal, T3 is loaded with the contents of the respective timer register (T2 or T4) and the respective service request SR0 or SR2 is activated.

Note: When a T3OTL transition is selected for the trigger signal, service request SR1 will also become active, indicating T3's overflow or underflow. Modifications of T3OTL via software will NOT trigger the counter function of T2/T4.

To ensure that a transition of the reload input signal applied to TxIN is recognized correctly, its level must be held high or low for a minimum number of module clock cycles, detailed in **Section 26.1.5**.

The Reload Mode triggered by the T3 toggle latch can be used in a number of different configurations. The following functions can be performed, depending on the selected active transition:

- If both a positive and a negative transition of T3OTL are selected to trigger a reload, the core timer will be reloaded with the contents of the auxiliary timer each time it overflows or underflows. This is the standard Reload Mode (reload on overflow/underflow).
- If either a positive or a negative transition of T3OTL is selected to trigger a reload, the core timer will be reloaded with the contents of the auxiliary timer on every second overflow or underflow.
- Using this “single-transition” mode for both auxiliary timers allows to perform very flexible Pulse Width Modulation (PWM). One of the auxiliary timers is programmed to reload the core timer on a positive transition of T3OTL, the other is programmed for a reload on a negative transition of T3OTL. With this combination the core timer is alternately reloaded from the two auxiliary timers.

Figure 26-16 shows an example for the generation of a PWM signal using the “single-transition” reload mechanism. T2 defines the high time of the PWM signal (reloaded on positive transitions) and T4 defines the low time of the PWM signal (reloaded on negative transitions). The PWM signal can be output on pin T3OUT if T3OE = 1. With this method, the high and low time of the PWM signal can be varied in a wide range.

Note: The output toggle latch T3OTL is accessible via software and may be changed, if required, to modify the PWM signal.
However, this will NOT trigger the reloading of T3.

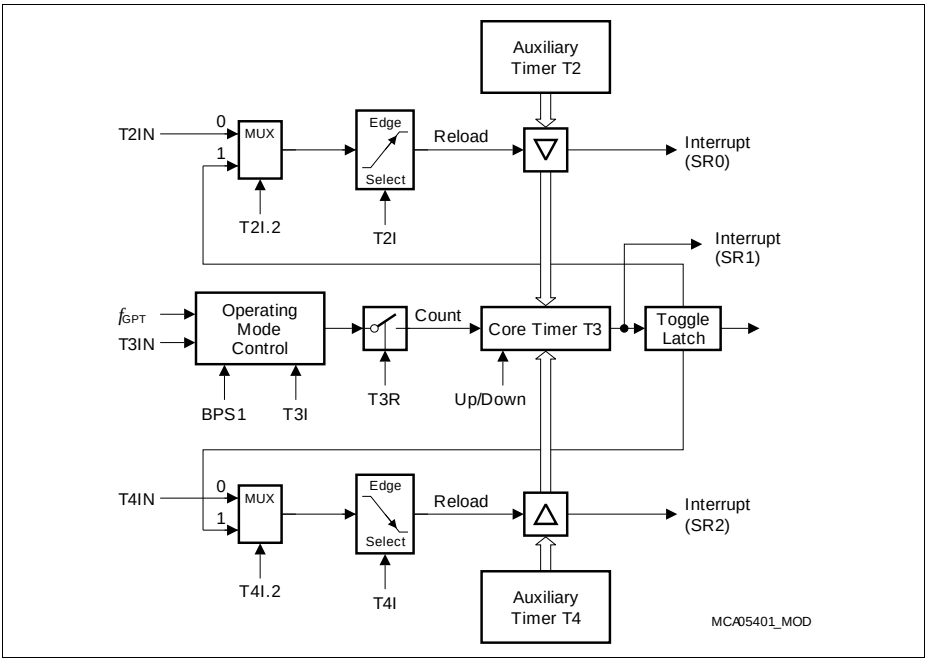


Figure 26-16 GPT1 Timer Reload Configuration for PWM Generation

Note: Although possible, selecting the same reload trigger event for both auxiliary timers should be avoided. In such a case, both reload registers would try to load the core timer at the same time. If this combination is selected, T2 is disregarded and the contents of T4 is reloaded.

The implementation of the GPT1 finite state machine may require special consideration in following cases.

In Case 1, when T2 or T4 are used to reload T3 on overflow/underflow, and T3 is read by software on the fly, the following unexpected values may be read from T3:

- When T3 is counting up, 0000_H or 0001_H may be read from T3 directly after an overflow, although the reload value in T2/T4 is higher (0001_H may be read in particular if BPS1 = 01_B and T3I = 000_B).
- When T3 is counting down, $FFFF_H$ or $FFFE_H$ may be read from T3 directly after an underflow, although the reload value in T2/T4 is lower ($FFFE_H$ may be read in particular if BPS1 = 01_B and T3I = 000_B).

Note: All timings derived from T3 in this configuration (e.g. distance between service requests, PWM waveform on T3OUT, etc.) are accurate except for the specific case described below.

Recommendation:

- When T3 counts up, and T3_value < reload value is read from T3, T3_value should be replaced with the reload value for further calculations.
- When T3 counts down, and T3_value > reload value is read from T3, T3_value should be replaced with the reload value for further calculations.

Alternatively, if the intention is to identify the overflow/underflow of T3, the T3 service request (SR1) may be used.

In Case 2, when T2 is used to reload T3 in the configuration with BPS1 = 01_B and T3I = 000_B (i.e. fastest configuration/highest resolution of T3), the reload of T3 is performed with a delay of one basic clock cycle.

Recommendation:

To compensate the delay and achieve correct timing,

- increment the reload value in T2 by 1 when T3 is configured to count up,
- decrement the reload value in T2 by 1 when T3 is configured to count down.

Alternatively, use T4 instead of T2 as reload register for T3. In this configuration the reload of T3 is not delayed, i.e. the effect described above does not occur with T4.

Timers T2 and T4 in Capture Mode

Capture Mode for an auxiliary timer Tx is selected by setting bitfield TxM in the respective register TxCON to 101_B. In Capture Mode, the contents of the core timer T3 are latched into an auxiliary timer register in response to a signal transition at the respective auxiliary timer's external input pin TxIN. The capture trigger signal can be a positive, a negative, or both a positive and a negative transition.

The two least significant bits of bitfield TxI select the active transition (see **Table 26-9**). Bit 2 of TxI is irrelevant for Capture Mode and must be cleared (TxI.2 = 0).

Note: When programmed for Capture Mode, the respective auxiliary timer (T2 or T4) stops independently of its run flag T2R or T4R.

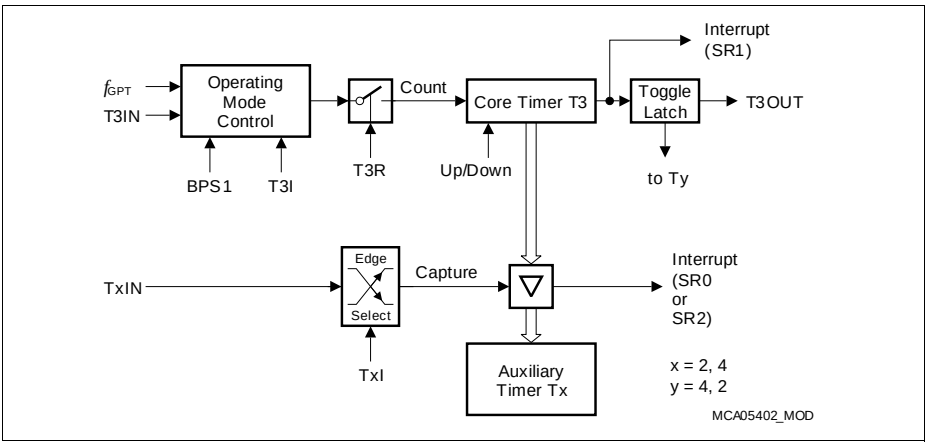


Figure 26-17 GPT1 Auxiliary Timer in Capture Mode

Upon a trigger (selected transition) at the corresponding input pin TxIN the contents of the core timer T3 are loaded into the auxiliary timer register Tx and the associated service request (SR0 or SR2) will be activated.

For Capture Mode operation, the respective timer input pin TxIN must be configured as input. To ensure that a transition of the capture input signal applied to TxIN is recognized correctly, its level must be held high or low for a minimum number of module clock cycles, detailed in **Section 26.1.5**.

26.1.5 GPT1 Clock Signal Control

All actions within the timer block GPT1 are triggered by transitions of its basic clock. This basic clock is derived from the module clock f_{GPT} by a basic block prescaler, controlled by bitfield BPS1 in register T3CON (see **Figure 26-1**). The count clock can be generated in two different ways:

- **Internal count clock**, derived from GPT1's basic clock via a programmable prescaler, is used for (Gated) Timer Mode.
- **External count clock**, derived from the timer's input pin(s), is used for Counter Mode or Incremental Interface Mode.

For both ways, the basic clock determines the maximum count frequency and the timer's resolution:

Table 26-3 Basic Clock Selection for Block GPT1

Block Prescaler[1)]	BPS1 = 01_B	BPS1 = 00_B[2)]	BPS1 = 11_B	BPS1 = 10_B
Prescaling Factor for GPT1: F(BPS1)	F(BPS1) = 4	F(BPS1) = 8	F(BPS1) = 16	F(BPS1) = 32
Maximum External Count Frequency	$f_{GPT}/8$	$f_{GPT}/16$	$f_{GPT}/32$	$f_{GPT}/64$
Input Signal Stable Time	4 x t_{GPT}	8 x t_{GPT}	16 x t_{GPT}	32 x t_{GPT}

1) Please note the non-linear encoding of bitfield BPS1.

2) Default after reset.

Note: The GPT1 block uses a finite state machine to control the actions. Since multiple interactions are possible between the timers (T2, T3, T4), these elements are processed sequentially. However, all actions are normally completed within one basic clock cycle. The GPT1 state machine has 8 states (4 states when BPS1 = 01_B) and processes the timers in the order T3 - T2 (all actions except capture) - T4 - T2 (capture).

Note: When initializing the GPT1 block after reset, and the block prescaler BPS1 in register T3CON needs to be set to a value different from its default value (00_B), it must be initialized first before any mode involving external trigger signals is configured. These modes include counter, incremental interface, capture, and reload mode. Otherwise, unintended count/capture/reload events may occur during the first basic clock cycle.
In this case, or when changing BPS1 during operation of the GPT1 block, disable related interrupts before modification of BPS1, and afterwards clear the corresponding service request flags and re-initialize those registers (T2, T3, T4) that might be affected by a count/capture/reload event.

Internal Count Clock Generation

In Timer Mode and Gated Timer Mode, the count clock for each GPT1 timer is derived from the GPT1 basic clock by a programmable prescaler, controlled by bitfield TxI in the respective timer's control register TxCON.

The count frequency f_{Tx} for a timer Tx and its resolution r_{Tx} are scaled linearly with lower clock frequencies, as can be seen from the following formula:

$$f_{Tx} = \frac{f_{GPT}}{F(BPS1) \times 2^{<TxI>}} \qquad r_{Tx}[\mu s] = \frac{F(BPS1) \times 2^{<TxI>}}{f_{GPT}[MHz]} \tag{26.1}$$

The effective count frequency depends on the common module clock prescaler factor F(BPS1) as well as on the individual input prescaler factor $2^{<TxI>}$. **Table 26-7** summarizes the resulting overall divider factors for a GPT1 timer that result from these cascaded prescalers.

Table 26-4 lists GPT1 timer's parameters (such as count frequency, resolution, and period) resulting from the selected overall prescaler factor and the module clock f_{GPT}. Note that some numbers may be rounded.

Table 26-4 GPT1 Timer Parameters

Module Clock f_{GPT} = 10 MHz			Overall Prescaler Factor	Module Clock f_{GPT} = 40 MHz		
Frequency	**Resolution**	**Period**		**Frequency**	**Resolution**	**Period**
2.5 MHz	400 ns	26.21 ms	4	10.0 MHz	100 ns	6.55 ms
1.25 MHz	800 ns	52.43 ms	8	5.0 MHz	200 ns	13.11 ms
625.0 kHz	1.6 µs	104.9 ms	16	2.5 MHz	400 ns	26.21 ms
312.5 kHz	3.2 µs	209.7 ms	32	1.25 MHz	800 ns	52.43 ms
156.25 kHz	6.4 µs	419.4 ms	64	625.0 kHz	1.6 µs	104.9 ms
78.125 kHz	12.8 µs	838.9 ms	128	312.5 kHz	3.2 µs	209.7 ms
39.06 kHz	25.6 µs	1.678 s	256	156.25 kHz	6.4 µs	419.4 ms
19.53 kHz	51.2 µs	3.355 s	512	78.125 kHz	12.8 µs	838.9 ms
9.77 kHz	102.4 µs	6.711 s	1024	39.06 kHz	25.6 µs	1.678 s
4.88 kHz	204.8 µs	13.42 s	2048	19.53 kHz	51.2 µs	3.355 s
2.44 kHz	409.6 µs	26.84 s	4096	9.77 kHz	102.4 µs	6.711 s

External Count Clock Input

The external input signals of the GPT1 block are sampled with the GPT1 basic clock (see **Figure 26-1**). To ensure that a signal is recognized correctly, its current level (high or

low) must be held active for at least one complete sampling period, before changing. A signal transition is recognized if two subsequent samples of the input signal represent different levels. Therefore, a minimum of two basic clock periods is required for the sampling of an external input signal. Thus, the maximum frequency of an input signal must not be higher than half the basic clock.

Table 26-5 summarizes the resulting requirements for external GPT1 input signals.

Table 26-5 GPT1 External Input Signal Limits

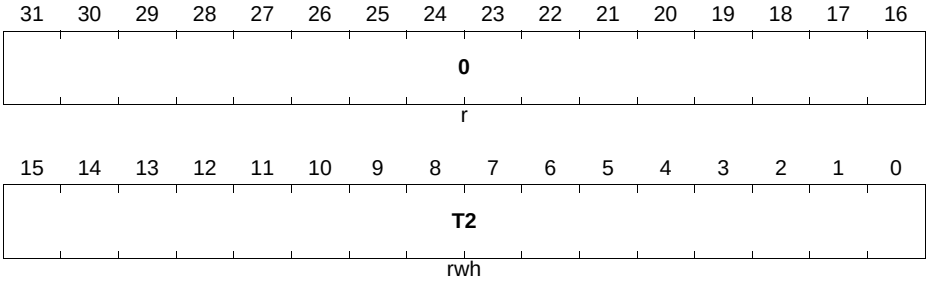
GPT1 Basic Clock = 10 MHz		Input Freq. Factor	GPT1 Divider BPS1	Input Phase Duration	GPT1 Basic Clock = 40 MHz	
Max. Input Frequency	**Min. Level Hold Time**				**Max. Input Frequency**	**Min. Level Hold Time**
1.25 MHz	400 ns	$f_{GPT}/8$	01_B	4 x t_{GPT}	5.0 MHz	100 ns
625.0 kHz	800 ns	$f_{GPT}/16$	00_B	8 x t_{GPT}	2.5 MHz	200 ns
312.5 kHz	1.6 µs	$f_{GPT}/32$	11_B	16 x t_{GPT}	1.25 MHz	400 ns
156.25 kHz	3.2 µs	$f_{GPT}/64$	10_B	32 x t_{GPT}	625.0 kHz	800 ns

These limitations are valid for all external input signals to GPT1, including the external count signals in Counter Mode and Incremental Interface Mode, the gate input signals in Gated Timer Mode, and the external direction signals.

26.1.6 GPT1 Registers

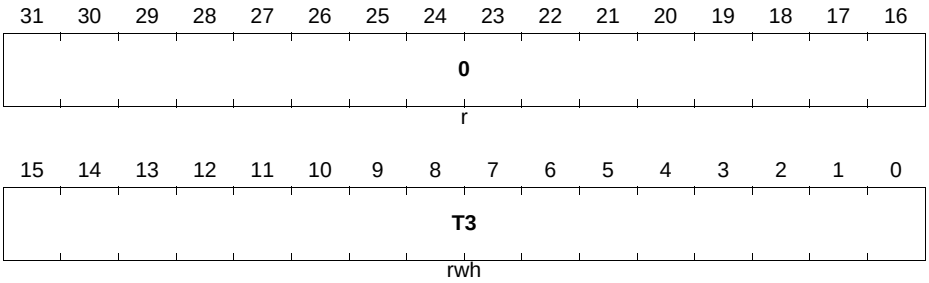
26.1.6.1 GPT1 Timer Registers

T2
Timer T2 Register **(34_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
T2	[15:0]	rwh	**Timer T2** Contains the current value of Timer T2.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

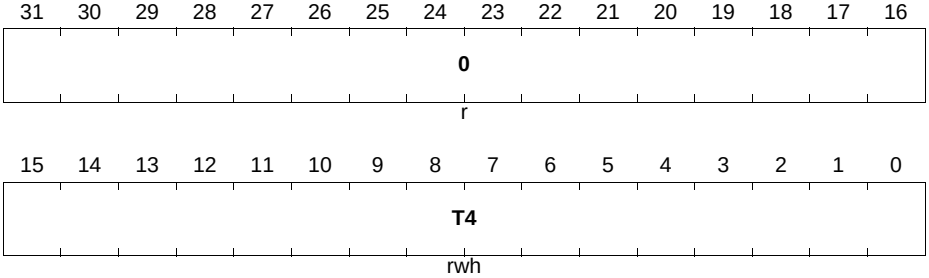
T3
Timer T3 Register **(38_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
T3	[15:0]	rwh	**Timer T3** Contains the current value of Timer T3.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

T4

Timer T4 Register **($3C_H$)** **Reset Value: 0000 0000$_H$**

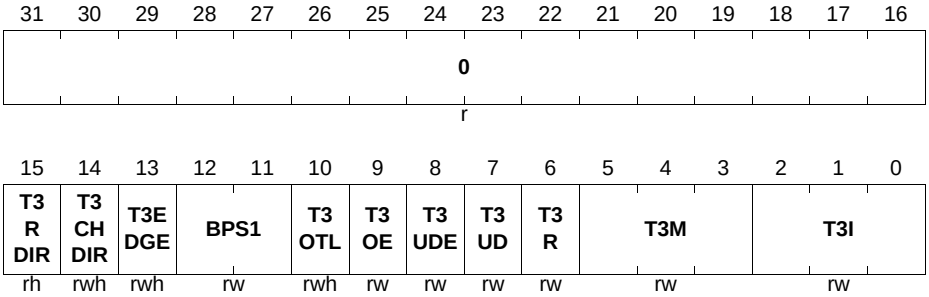


Field	Bits	Type	Description
T4	[15:0]	rwh	**Timer T4** Contains the current value of Timer T4.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

26.1.6.2 GPT1 Timer Control Registers

GPT1 Core Timer T3 Control Register

T3CON
Timer T3 Control Register **(14_H)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
T3I	[2:0]	rw	**Timer T3 Input Parameter Selection** Depends on the operating mode, see respective sections for encoding: **Table 26-7** for Timer Mode and Gated Timer Mode **Table 26-8** for Counter Mode **Table 26-10** for Incremental Interface Mode
T3M	[5:3]	rw	**Timer T3 Mode Control** 000_B Timer Mode 001_B Counter Mode 010_B Gated Timer Mode with gate active low 011_B Gated Timer Mode with gate active high 100_B Reserved. Do not use this combination 101_B Reserved. Do not use this combination 110_B Incremental Interface Mode (Rotation Detection Mode) 111_B Incremental Interface Mode (Edge Detection Mode)
T3R	6	rw	**Timer T3 Run Bit** 0_B Timer T3 stops 1_B Timer T3 runs

Field	Bits	Type	Description
T3UD	7	rw	**Timer T3 Up/Down Control**[1)] 0_B Timer T3 counts up 1_B Timer T3 counts down *Note: This bit only controls count direction of T3 if bit T3UDE = 0.*
T3UDE	8	rw	**Timer T3 External Up/Down Enable**[1)] 0_B Count direction is controlled by bit T3UD; input T3EUD is disconnected 1_B Count direction is controlled by input T3EUD
T3OE	9	rw	**Overflow/Underflow Output Enable** 0_B Alternate Output Function Disabled 1_B State of T3 toggle latch is output on pin T3OUT
T3OTL	10	rwh	**Timer T3 Overflow Toggle Latch** Toggles on each overflow/underflow of T3. Can be set or cleared by software (see separate description)
BPS1	[12:11]	rw	**GPT1 Block Prescaler Control** Selects the basic clock for block GPT1 (see also **Section 26.1.5**) 00_B $f_{GPT}/8$ 01_B $f_{GPT}/4$ 10_B $f_{GPT}/32$ 11_B $f_{GPT}/16$
T3EDGE	13	rwh	**Timer T3 Edge Detection Flag** The bit is set each time a count edge is detected. T3EDGE must be cleared by software. 0_B No count edge was detected 1_B A count edge was detected
T3CHDIR	14	rwh	**Timer T3 Count Direction Change Flag** This bit is set each time the count direction of timer T3 changes. T3CHDIR must be cleared by software. 0_B No change of count direction was detected 1_B A change of count direction was detected
T3RDIR	15	rh	**Timer T3 Rotation Direction Flag** 0_B Timer T3 counts up 1_B Timer T3 counts down
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

1) See **Table 26-6** for encoding of bits T3UD and T3UDE.

GPT1 Auxiliary Timers T2/T4 Control Registers

T2CON
Timer T2 Control Register **(10_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T2R DIR	T2CH DIR	T2EDGE	T2IRDIS	0		T2RC	T2UDE	T2UD	T2R	T2M			T2I		
rh	rwh	rwh	rw	r		rw	rw	rw	rw	rw			rw		

Field	Bits	Type	Description
T2I	[2:0]	rw	**Timer T2 Input Parameter Selection** Depends on the operating mode, see respective sections for encoding: **Table 26-7** for Timer Mode and Gated Timer Mode **Table 26-9** for Counter Mode **Table 26-10** for Incremental Interface Mode
T2M	[5:3]	rw	**Timer T2 Mode Control (Basic Operating Mode)** 000_B Timer Mode 001_B Counter Mode 010_B Gated Timer Mode with gate active low 011_B Gated Timer Mode with gate active high 100_B Reload Mode 101_B Capture Mode 110_B Incremental Interface Mode (Rotation Detection Mode) 111_B Incremental Interface Mode (Edge Detection Mode)
T2R	6	rw	**Timer T2 Run Bit** 0_B Timer T2 stops 1_B Timer T2 runs *Note: This bit only controls timer T2 if bit T2RC = 0.*

Field	Bits	Type	Description
T2UD	7	rw	**Timer T2 Up/Down Control**[1)] 0_B Timer T2 counts up 1_B Timer T2 counts down *Note: This bit only controls count direction of T2 if bit T2UDE = 0.*
T2UDE	8	rw	**Timer T2 External Up/Down Enable**[1)] 0_B Count direction is controlled by bit T2UD; input T2EUD is disconnected 1_B Count direction is controlled by input T2EUD
T2RC	9	rw	**Timer T2 Remote Control** 0_B Timer T2 is controlled by its own run bit T2R 1_B Timer T2 is controlled by the run bit T3R of core timer T3, not by bit T2R
T2IRDIS	12	rw	**Timer T2 Interrupt Disable** 0_B Interrupt generation for T2CHDIR and T2EDGE interrupts in Incremental Interface Mode is enabled 1_B Interrupt generation for T2CHDIR and T2EDGE interrupts in Incremental Interface Mode is disabled
T2EDGE	13	rwh	**Timer T2 Edge Detection** The bit is set each time a count edge is detected. T2EDGE must be cleared by software. 0_B No count edge was detected 1_B A count edge was detected
T2CHDIR	14	rwh	**Timer T2 Count Direction Change** The bit is set each time the count direction of timer T2 changes. T2CHDIR must be cleared by software. 0_B No change in count direction was detected 1_B A change in count direction was detected
T2RDIR	15	rh	**Timer T2 Rotation Direction** 0_B Timer T2 counts up 1_B Timer T2 counts down
0	[11:10], [31:16]	r	**Reserved** Read as 0; should be written with 0.

1) See **Table 26-6** for encoding of bits T2UD and T2UDE.

T4CON
Timer T4 Control Register **(18_H)** **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T4R DIR	T4CH DIR	T4EDGE	T4IRDIS	CLRT3EN	CLRT2EN	T4RC	T4UDE	T4UD	T4R	T4M			T4I		
rh	rwh	rwh	rw	rw	rw	rw	rw	rw	rw	rw			rw		

Field	Bits	Type	Description
T4I	[2:0]	rw	**Timer T4 Input Parameter Selection** Depends on the operating mode, see respective sections for encoding: **Table 26-7** for Timer Mode and Gated Timer Mode **Table 26-9** for Counter Mode **Table 26-10** for Incremental Interface Mode
T4M	[5:3]	rw	**Timer T4 Mode Control (Basic Operating Mode)** 000_B Timer Mode 001_B Counter Mode 010_B Gated Timer Mode with gate active low 011_B Gated Timer Mode with gate active high 100_B Reload Mode 101_B Capture Mode 110_B Incremental Interface Mode (Rotation Detection Mode) 111_B Incremental Interface Mode (Edge Detection Mode)
T4R	6	rw	**Timer T4 Run Bit** 0_B Timer T4 stops 1_B Timer T4 runs *Note: This bit only controls timer T4 if bit T4RC = 0.*

Field	Bits	Type	Description
T4UD	7	rw	**Timer T4 Up/Down Control[1)]** 0_B Timer T4 counts up 1_B Timer T4 counts down *Note: This bit only controls count direction of T4 if bit T4UDE = 0.*
T4UDE	8	rw	**Timer T4 External Up/Down Enable[1)]** 0_B Count direction is controlled by bit T4UD; input T4EUD is disconnected 1_B Count direction is controlled by input T4EUD
T4RC	9	rw	**Timer T4 Remote Control** 0_B Timer T4 is controlled by its own run bit T4R 1_B Timer T4 is controlled by the run bit T3R of core timer T3, but not by bit T4R
CLRT2EN	10	rw	**Clear Timer T2 Enable** Enables the automatic clearing of timer T2 upon a falling edge of the selected T4EUD input. 0_B No effect of T4EUD on timer T2 1_B A falling edge on T4EUD clears timer T2
CLRT3EN	11	rw	**Clear Timer T3 Enable** Enables the automatic clearing of timer T3 upon a falling edge of the selected T4IN input. 0_B No effect of T4IN on timer T3 1_B A falling edge on T4IN clears timer T3
T4IRDIS	12	rw	**Timer T4 Interrupt Disable** 0_B Interrupt generation for T4CHDIR and T4EDGE interrupts in Incremental Interface Mode is enabled 1_B Interrupt generation for T4CHDIR and T4EDGE interrupts in Incremental Interface Mode is disabled
T4EDGE	13	rwh	**Timer T4 Edge Detection** The bit is set each time a count edge is detected. T4EDGE has to be cleared by software. 0_B No count edge was detected 1_B A count edge was detected

Field	Bits	Type	Description
T4CHDIR	14	rwh	**Timer T4 Count Direction Change** The bit is set each time the count direction of timer T4 changes. T4CHDIR must be cleared by software. 0_B No change in count direction was detected 1_B A change in count direction was detected
T4RDIR	15	rh	**Timer T4 Rotation Direction** 0_B Timer T4 counts up 1_B Timer T4 counts down
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

1) See **Table 26-6** for encoding of bits T4UD and T4UDE.

Encoding of GPT1 Timer Count Direction Control

Table 26-6 GPT1 Timer Count Direction Control

Pin TxEUD	Bit TxUDE	Bit TxUD	Count Direction	Bit TxRDIR
X	0	0	Count Up	0
X	0	1	Count Down	1
0	1	0	Count Up	0
1	1	0	Count Down	1
0	1	1	Count Down	1
1	1	1	Count Up	0

Encoding of GPT1 Overall Prescaler Factor in Timer Mode and Gated Timer Mode

Table 26-7 GPT1 Overall Prescaler Factors for Internal Count Clock (Timer Mode and Gated Timer Mode)

Individual Prescaler for Tx	Common Prescaler for Module Clock[1]			
	BPS1 = 01_B	BPS1 = 00_B	BPS1 = 11_B	BPS1 = 10_B
TxI = 000_B	4	8	16	32
TxI = 001_B	8	16	32	64
TxI = 010_B	16	32	64	128
TxI = 011_B	32	64	128	256
TxI = 100_B	64	128	256	512
TxI = 101_B	128	256	512	1024
TxI = 110_B	256	512	1024	2048
TxI = 111_B	512	1024	2048	4096

1) Please note the non-linear encoding of bitfield BPS1.

Encoding of GPT1 Input Edge Selection in Counter Mode

Table 26-8 GPT1 Core Timer T3 Input Edge Selection (Counter Mode)

TxI	Triggering Edge for Counter Increment/Decrement
000_B	None. Counter T3 is disabled
001_B	Positive transition (rising edge) on T3IN
010_B	Negative transition (falling edge) on T3IN
011_B	Any transition (rising or falling edge) on T3IN
$1XX_B$	Reserved. Do not use this combination

Table 26-9 GPT1 Auxiliary Timers T2/T4 Input Edge Selection

T2I/T4I	Triggering Edge for Counter Increment/Decrement, Capture[1] or Reload
$X00_B$	None. Counter Tx is disabled
001_B	Positive transition (rising edge) on TxIN
010_B	Negative transition (falling edge) on TxIN
011_B	Any transition (rising or falling edge) on TxIN
101_B	Positive transition (rising edge) of T3 toggle latch T3OTL[2]
110_B	Negative transition (falling edge) of T3 toggle latch T3OTL[2]
111_B	Any transition (rising or falling edge) of T3 toggle latch T3OTL[2]

1) For Capture Mode, only settings TxI = 0XX may be used.

2) Not for Capture Mode.

Encoding of Input Edge Selection in Incremental Interface Mode

Table 26-10 GPT1 Timer Tx Input Edge Selection (Incremental Interface Mode)

TxI	Triggering Edge for Counter Increment/Decrement
000_B	None. Counter Tx stops.
001_B	Any transition (rising or falling edge) on TxIN.
010_B	Any transition (rising or falling edge) on TxEUD.
011_B	Any transition (rising or falling edge) on any Tx input (TxIN or TxEUD).
$1XX_B$	Reserved. Do not use this combination.

26.2 Timer Block GPT2

Both timers of block GPT2 (T5, T6) can run in one of 3 basic modes: Timer Mode, Gated Timer Mode, or Counter Mode. All timers can count up or down. Each timer of GPT2 is controlled by a separate control register TxCON.

Each timer has an input pin TxIN (alternate pin function) associated with it, which serves as the gate control in Gated Timer Mode, or as the count input in Counter Mode. The count direction (up/down) may be programmed via software or may be dynamically altered by a signal at the External Up/Down control input TxEUD (alternate pin function). An overflow/underflow of core timer T6 is indicated by the Output Toggle Latch T6OTL, whose state may be output on the associated pin T6OUT (alternate pin function). The auxiliary timer T5 may additionally be concatenated with core timer T6 (through T6OTL).

The Capture/Reload register CAPREL can be used to capture the contents of timer T5, or to reload timer T6. A special mode facilitates the use of register CAPREL for both functions at the same time. This mode allows frequency multiplication. The capture function is triggered by the input pin CAPIN, or by GPT1 timer's T3 input lines T3IN and T3EUD. The reload function is triggered by an overflow or underflow of timer T6.

The current contents of each timer can be read or modified by the CPU by accessing the corresponding timer count registers T5 or T6. When any of the timer registers is written by the CPU in the state immediately preceding a timer increment, decrement, reload, or capture operation, the CPU write operation has priority in order to guarantee correct results.

The interrupt requests of GPT2 are signalled on service request lines SR3, SR4, and SR5.

The input and output lines of GPT2 are connected to pins. The control registers for the port functions are located in the respective port modules.

*Note: The timing requirements for external input signals can be found in **Section 26.2.6**, **Section 26.5.2** summarizes the module interface signals, including pins and interrupt request signals.*

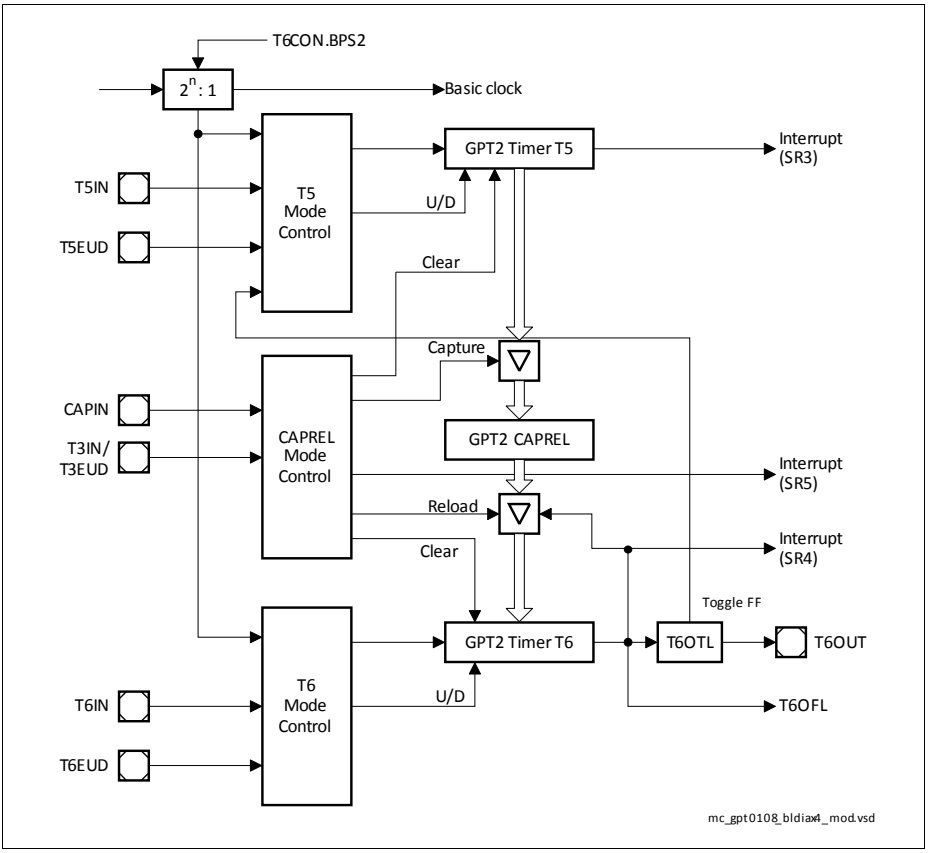


Figure 26-18 GPT2 Block Diagram

26.2.1 GPT2 Core Timer T6 Control

The current contents of the core timer T6 are reflected by its count register T6. This register can also be written to by the CPU, for example, to set the initial start value.

The core timer T6 is configured and controlled via its control register T6CON.

Timer T6 Run Control

The core timer T6 can be started or stopped by software through bit T6R (timer T6 run bit). This bit is relevant in all operating modes of T6. Setting bit T6R will start the timer, clearing bit T6R stops the timer.

In Gated Timer Mode, the timer will only run if T6R = 1 and the gate is active (high or low, as programmed).

Note: When bit T5RC in timer control register T5CON is set, bit T6R will also control (start and stop) the Auxiliary Timer T5.

Count Direction Control

The count direction of the GPT2 timers (core timer and auxiliary timer) can be controlled either by software or by the external input pin TxEUD (Timer Tx External Up/Down Control Input). These options are selected by bits TxUD and TxUDE in the respective control register TxCON. When the up/down control is provided by software (bit TxUDE = 0), the count direction can be altered by setting or clearing bit TxUD. When bit TxUDE = 1, pin TxEUD is selected to be the controlling source of the count direction. However, bit TxUD can still be used to reverse the actual count direction, as shown in **Table 26-15**. The count direction can be changed regardless of whether or not the timer is running.

Timer T6 Output Toggle Latch

The overflow/underflow signal of timer T6 is connected to a block named ‘Toggle Latch’, shown in the Timer Mode diagrams. **Figure 26-19** illustrates the details of this block. An overflow or underflow of T6 will clock two latches: The first latch represents bit T6OTL in control register T6CON. The second latch is an internal latch toggled by T6OTL’s output. Both latch outputs are connected to the input control block of the auxiliary timer T5. The output level of the shadow latch will match the output level of T6OTL, but is delayed by one clock cycle. When the T6OTL value changes, this will result in a temporarily different output level from T6OTL and the shadow latch, which can trigger the selected count event in T5.

When software writes to T6OTL, both latches are set or cleared simultaneously. In this case, both signals to the auxiliary timer carry the same level and no edge will be detected. Bit T6OE (overflow/underflow output enable) in register T6CON enables the state of T6OTL to be monitored via an external pin T6OUT. When T6OTL is linked to an external port pin (must be configured as output), T6OUT can be used to control external HW. If T6OE = 1, pin T6OUT outputs the state of T6OTL. If T6OE = 0, pin T6OUT outputs a high level (while it selects the timer output signal).

As can be seen from **Figure 26-19**, when latch T6OTL is modified by software to determine the state of the output line, also the internal shadow latch is set or cleared accordingly. Therefore, no trigger condition is detected by T5 in this case.

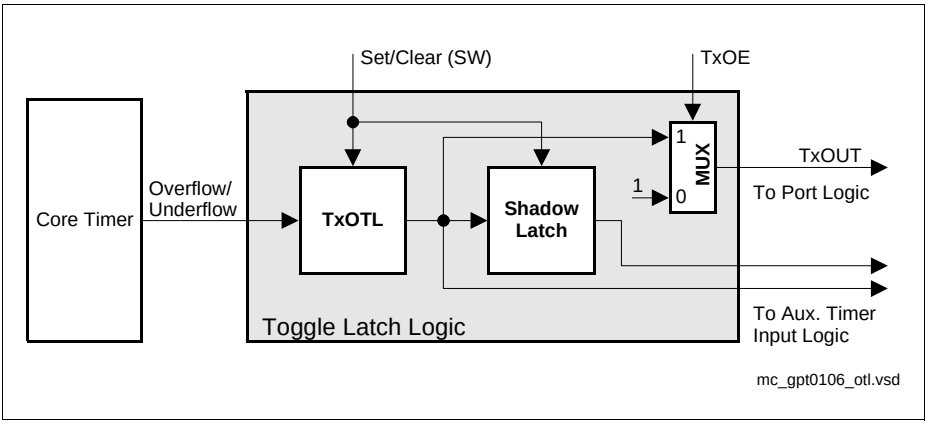


Figure 26-19 Block Diagram of the Toggle Latch Logic of Core Timer T6 (x = 6)

26.2.2 GPT2 Core Timer T6 Operating Modes

Timer T6 can operate in one of several modes.

Timer T6 in Timer Mode

Timer Mode for the core timer T6 is selected by setting bitfield T6M in register T6CON to 000_B. In this mode, T6 is clocked with the module's input clock f_{GPT} divided by two programmable prescalers controlled by bitfields BPS2 and T6I in register T6CON. Please see **Section 26.2.6** for details on the input clock options.

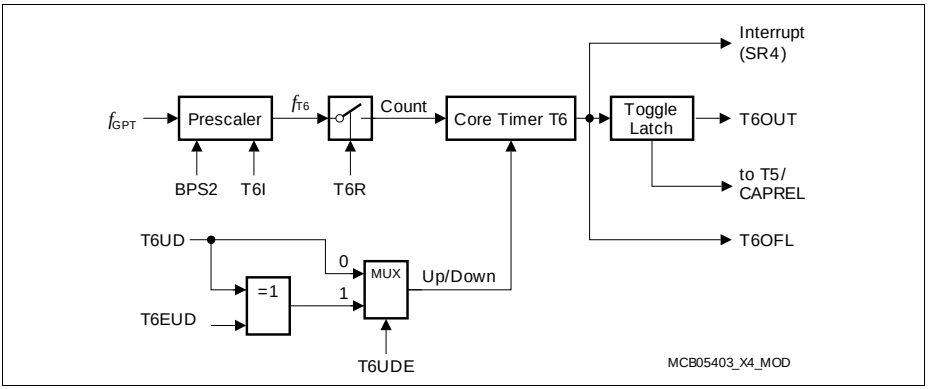


Figure 26-20 Block Diagram of Core Timer T6 in Timer Mode

Timer 6 in Gated Timer Mode

Gated Timer Mode for the core timer T6 is selected by setting bitfield T6M in register T6CON to 010_B or 011_B. Bit T6M.0 (T6CON.3) selects the active level of the gate input. The same options for the input frequency are available in Gated Timer Mode as in Timer Mode (see **Section 26.2.6**). However, the input clock to the timer in this mode is gated by the external input pin T6IN (Timer T6 External Input).
To enable this operation, the associated pin T6IN must be configured as input.

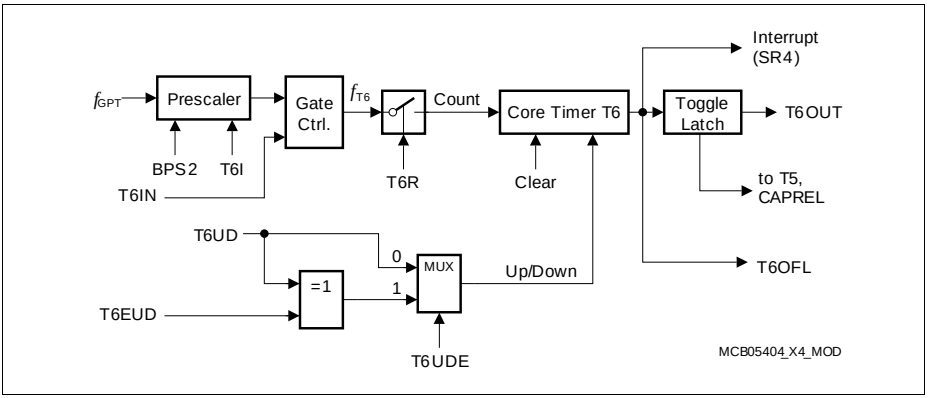


Figure 26-21 Block Diagram of Core Timer T6 in Gated Timer Mode

If T6M = 010_B, the timer is enabled when T6IN shows a low level. A high level at this line stops the timer. If T6M = 011_B, line T6IN must have a high level in order to enable the timer. Additionally, the timer can be turned on or off by software using bit T6R. The timer will only run if T6R is 1 and the gate is active. It will stop if either T6R is 0 or the gate is inactive.

Note: A transition of the gate signal at pin T6IN does not cause a service request.

Timer 6 in Counter Mode

Counter Mode for the core timer T6 is selected by setting bitfield T6M in register T6CON to 001_B. In Counter Mode, timer T6 is clocked by a transition at the external input pin T6IN. The event causing an increment or decrement of the timer can be a positive, a negative, or both a positive and a negative transition at this line. Bitfield T6I in control register T6CON selects the triggering transition (see **Table 26-17**).

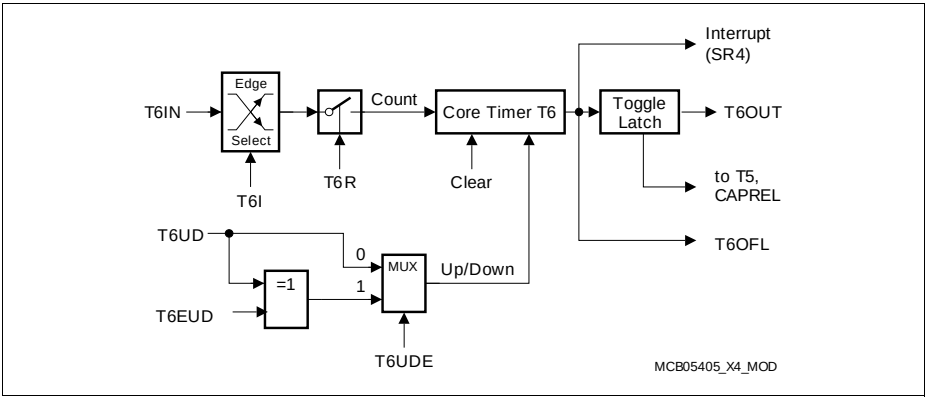


Figure 26-22 Block Diagram of Core Timer T6 in Counter Mode

For Counter Mode operation, pin T6IN must be configured as input. The maximum input frequency allowed in Counter Mode depends on the selected prescaler value. To ensure that a transition of the count input signal applied to T6IN is recognized correctly, its level must be held high or low for a minimum number of module clock cycles before it changes. This information can be found in **Section 26.2.6**.

26.2.3 GPT2 Auxiliary Timer T5 Control

Auxiliary timer T5 can be configured for Timer Mode, Gated Timer Mode, or Counter Mode with the same options for the timer frequencies and the count signal as the core timer T6. In addition to these 3 counting modes, the auxiliary timer can be concatenated with the core timer. The contents of T5 may be captured to register CAPREL upon an external or an internal trigger. The start/stop function of the auxiliary timer can be remotely controlled by the T6 run control bit. Both timers may thus be controlled synchronously.

The current contents of the auxiliary timer are reflected by its count register T5. This register can also be written to by the CPU, for example, to set the initial start value.

The individual configurations for timer T5 are determined by its control register T5CON. Some bits in this register also control the function of the CAPREL register. Note that functions which are present in all timers of block GPT2 are controlled in the same bit positions and in the same manner in each of the specific control registers.

Note: The auxiliary timer has no output toggle latch and no alternate output function.

Timer T5 Run Control

The auxiliary timer T5 can be started or stopped by software in two different ways:

- Through the associated timer run bit (T5R). In this case it is required that the respective control bit T5RC = 0.
- Through the core timer's run bit (T6R). In this case the respective remote control bit must be set (T5RC = 1).

The selected run bit is relevant in all operating modes of T5. Setting the bit will start the timer, clearing the bit stops the timer.

In Gated Timer Mode, the timer will only run if the selected run bit is set and the gate is active (high or low, as programmed).

Note: If remote control is selected T6R will start/stop timer T6 and the auxiliary timer T5 synchronously.

26.2.4 GPT2 Auxiliary Timer T5 Operating Modes

The operation of the auxiliary timer in the basic operating modes is almost identical with the core timer's operation, with very few exceptions. Additionally, some combined operating modes can be selected.

Timer T5 in Timer Mode

Timer Mode for the auxiliary timer T5 is selected by setting its bitfield T5M in register T5CON to 000_B.

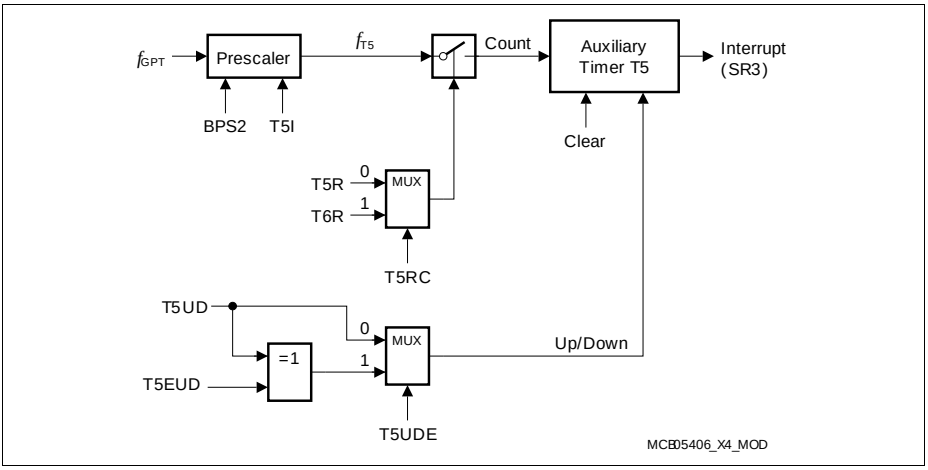


Figure 26-23 Block Diagram of Auxiliary Timer T5 in Timer Mode

Timer T5 in Gated Timer Mode

Gated Timer Mode for the auxiliary timer T5 is selected by setting bitfield T5M in register T5CON to 010_B or 011_B. Bit T5M.0 (T5CON.3) selects the active level of the gate input.

Note: A transition of the gate signal at line T5IN does not cause a service request.

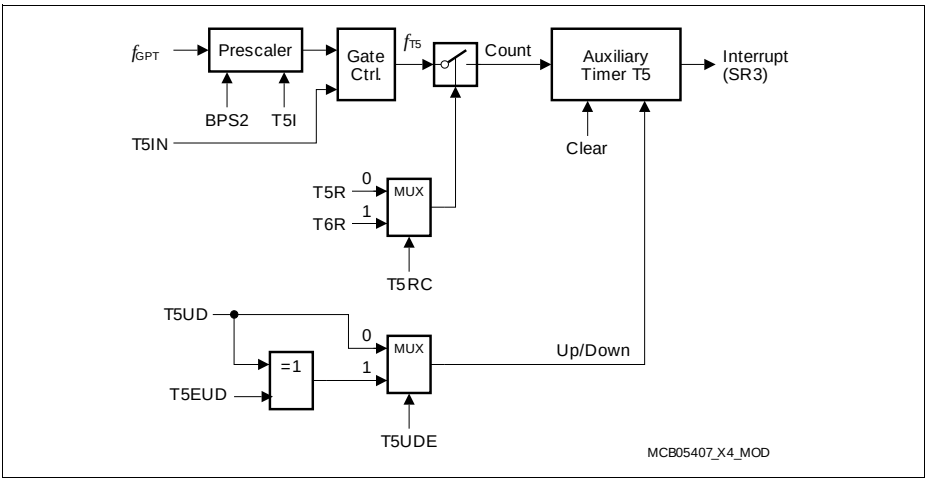


Figure 26-24 Block Diagram of Auxiliary Timer T5 in Gated Timer Mode

Note: There is no output toggle latch for T5.
Start/stop of the auxiliary timer can be controlled locally or remotely.

Timer T5 in Counter Mode

Counter Mode for auxiliary timer T5 is selected by setting bitfield T5M in register T5CON to 001_B. In Counter Mode, the auxiliary timer can be clocked either by a transition at its external input line T5IN, or by a transition of timer T6's toggle latch T6OTL. The event causing an increment or decrement of a timer can be a positive, a negative, or both a positive and a negative transition at either the respective input pin or at the toggle latch. Bitfield T5I in control register T5CON selects the triggering transition (see **Table 26-18**).

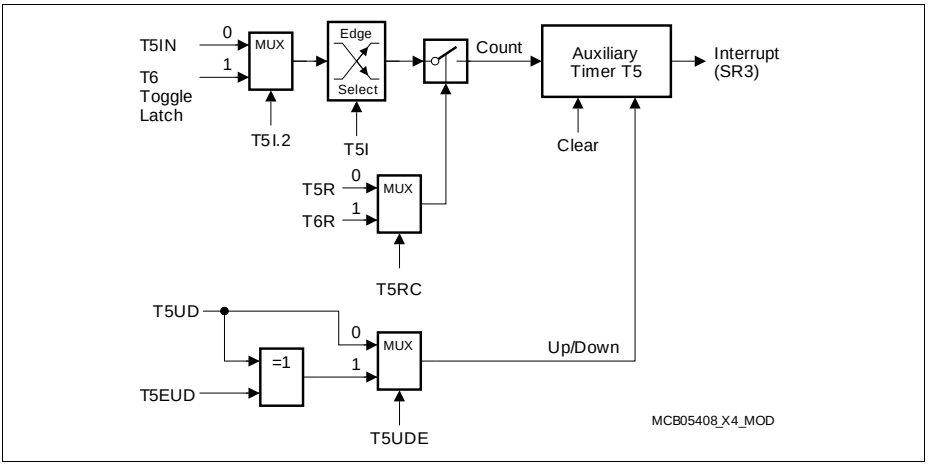


Figure 26-25 Block Diagram of Auxiliary Timer T5 in Counter Mode

Note: Only state transitions of T6OTL which are caused by the overflows/underflows of T6 will trigger the counter function of T5. Modifications of T6OTL via software will NOT trigger the counter function of T5.

For counter operation, pin T5IN must be configured as input. The maximum input frequency allowed in Counter Mode depends on the selected prescaler value. To ensure that a transition of the count input signal applied to T5IN is recognized correctly, its level must be held high or low for a minimum number of module clock cycles before it changes. This information can be found in **Section 26.2.6**.

Timer Concatenation

Using the toggle bit T6OTL as a clock source for the auxiliary timer in Counter Mode concatenates the core timer T6 with the auxiliary timer T5. This concatenation forms either a 32-bit or a 33-bit timer/counter, depending on which transition of T6OTL is selected to clock the auxiliary timer.

- **32-bit Timer/Counter:** If both a positive and a negative transition of T6OTL are used to clock the auxiliary timer, this timer is clocked on every overflow/underflow of the core timer T6. Thus, the two timers form a 32-bit timer.
- **33-bit Timer/Counter:** If either a positive or a negative transition of T6OTL is selected to clock the auxiliary timer, this timer is clocked on every second overflow/underflow of the core timer T6. This configuration forms a 33-bit timer (16-bit core timer + T6OTL + 16-bit auxiliary timer).
  As long as bit T6OTL is not modified by software, it represents the state of the internal toggle latch, and can be regarded as part of the 33-bit timer.

The count directions of the two concatenated timers are not required to be the same. This offers a wide variety of different configurations.

T6, which represents the low-order part of the concatenated timer, can operate in Timer Mode, Gated Timer Mode or Counter Mode in this case.

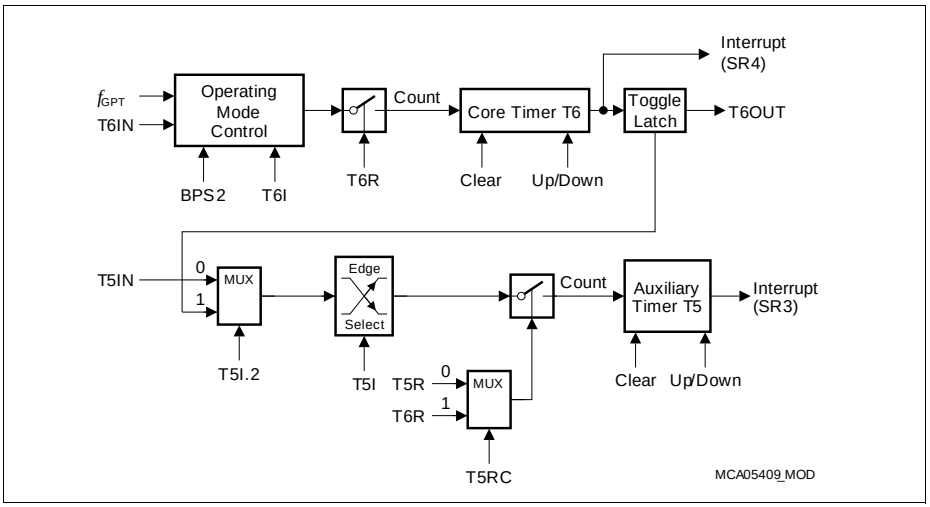


Figure 26-26 Concatenation of Core Timer T6 and Auxiliary Timer T5

When reading the low and high parts of the concatenated timer, care must be taken to obtain consistent values in particular after a timer overflow/underflow (e.g. one part may already have considered an overflow, while the other has not).

Note: This is a general issue when reading multi-word results with consecutive instructions, and not necessarily unique to the GPT12 module architecture.

The following algorithm may be used to read concatenated GPT2 timers, represented by TIMER_HIGH (for auxiliary timer T5) and TIMER_LOW (for core timer T6). The high part is read twice, and reading of the low part is repeated if two different values were read for the high part:

- TIMER_HIGH_TMP = T5
- TIMER_LOW = T6
- Wait two basic clock cycles (to allow increment/decrement of auxiliary timer in case of core timer overflow/underflow) - see **Table 26-11**
- TIMER_HIGH = T5
- If TIMER_HIGH is not equal to TIMER_HIGH_TMP then TIMER_LOW = T6

After execution of this algorithm, TIMER_HIGH and TIMER_LOW represent a consistent time stamp of the concatenated timers.

The equivalent number of module clock cycles corresponding to two basic clock cycles is shown in **Table 26-11**.

Table 26-11 Number of Module Clock Cycles to Wait for Two Basic Clock Cycles

Block Prescaler	BPS2 = 01_B	BPS2 = 00_B	BPS2 = 11_B	BPS2 = 10_B
Number of module clocks	4	8	16	32

In case the required timer resolution can be achieved with different combinations of the Block Prescaler BPS2 and the Individual Prescalers TxI (see **Table 26-16**), the variant with the smallest value for the Block Prescaler may be chosen to minimize the waiting time. E.g. in order to run T6 at f_{SYS}/512, select BPS2 = 00_B, T6I = 111_B, and insert 8 NOPs (or other instructions) to ensure the required waiting time before reading TIMER_HIGH the second time.

26.2.5 GPT2 Register CAPREL Operating Modes

The Capture/Reload register CAPREL can be used to capture the contents of timer T5, or to reload timer T6. A special mode facilitates the use of register CAPREL for both functions at the same time. This mode allows frequency multiplication. The capture function is triggered by the input pin CAPIN, by GPT1 timer's T3 input lines T3IN and T3EUD, or by read accesses to GPT1 timers. The reload function is triggered by an overflow or underflow of timer T6.

The capture trigger signal can also be used to clear the contents of timers T5 and T6 individually.

The functions of register CAPREL are controlled via several bit(field)s in the timer control registers T5CON and T6CON.

Capture/Reload Register CAPREL in Capture Mode

Capture Mode for register CAPREL is selected by setting bit T5SC in control register T5CON (set bitfield CI in register T5CON to a non-zero value to select a trigger signal). In Capture Mode, the contents of the auxiliary timer T5 are latched into register CAPREL in response to a signal transition at the selected external input pin(s). Bit CT3 selects the external input line CAPIN or the input lines T3IN and/or T3EUD of GPT1 timer T3 as the source for a capture trigger. Either a positive, a negative, or both a positive and a negative transition at line CAPIN can be selected to trigger the capture function, or transitions on input T3IN or input T3EUD or both inputs, T3IN and T3EUD. The active edge is controlled by bitfield CI in register T5CON. **Table 26-19** summarizes these options.

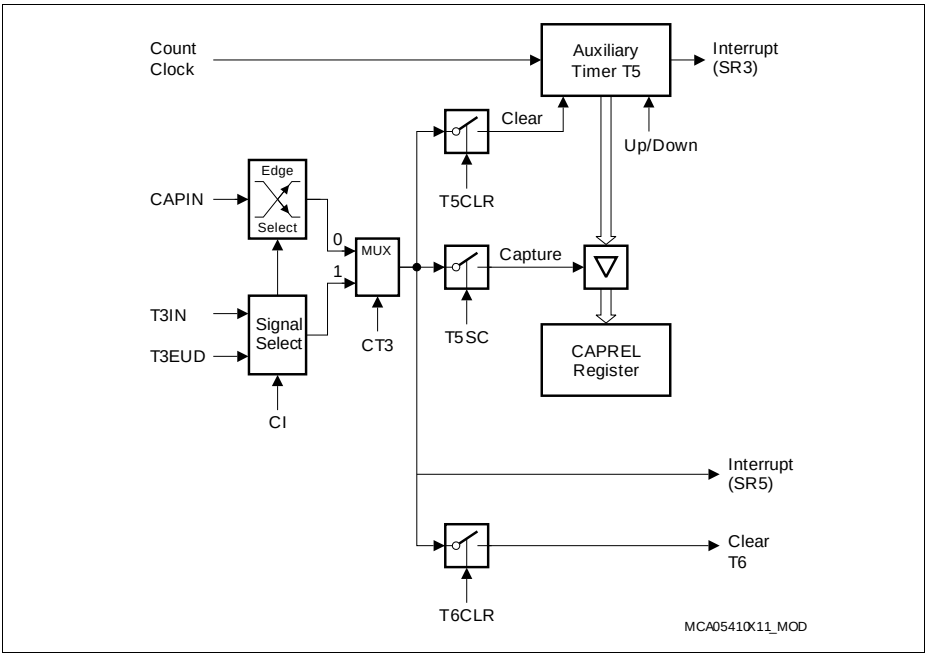


Figure 26-27 Capture/Reload Register CAPREL in Capture Mode

When a selected trigger is detected, the contents of the auxiliary timer T5 are latched into register CAPREL and the service request is activated. The same event can optionally clear timer T5 and/or timer T6. This option is enabled by bit T5CLR in register T5CON and bit T6CLR in register T6CON, respectively. If TxCLR = 0 the contents of timer Tx is not affected by a capture. If TxCLR = 1 timer Tx is cleared after the current timer T5 value has been latched into register CAPREL.

Note: Bit T5SC only controls whether or not a capture is performed. If T5SC is cleared the external input pin(s) can still be used to clear timer T5 and/or T6, or as external interrupt input(s). This interrupt is signalled by the CAPREL interrupt request SR5.

When capture triggers T3IN or T3EUD are enabled (CT3 = 1), register CAPREL captures the contents of T5 upon transitions of the selected input(s). These values can be used to measure T3's input signals. This is useful, for example, when T3 operates in Incremental Interface Mode, in order to derive dynamic information (speed, acceleration) from the input signals.

For Capture Mode operation, the selected pins CAPIN, T3IN, or T3EUD must be configured as input. To ensure that a transition of a trigger input signal applied to one of

these inputs is recognized correctly, its level must be held high or low for a minimum number of module clock cycles, detailed in **Section 26.2.6**.

Capture/Reload Register CAPREL in Reload Mode

Reload Mode for register CAPREL is selected by setting bit T6SR in control register T6CON. In Reload Mode, the core timer T6 is reloaded with the contents of register CAPREL, triggered by an overflow or underflow of T6. This will not activate the service request SR5 associated with the CAPREL register. However, service request SR4 will be activated, indicating the overflow/underflow of T6.

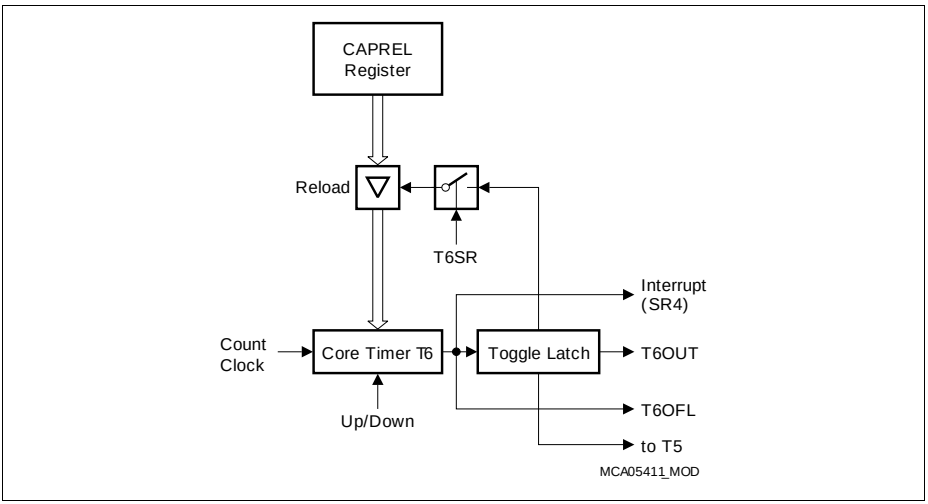


Figure 26-28 Capture/Reload Register CAPREL in Reload Mode

Capture/Reload Register CAPREL in Capture-And-Reload Mode

Since the reload function and the capture function of register CAPREL can be enabled individually by bits T5SC and T6SR, the two functions can be enabled simultaneously by setting both bits. This feature can be used to generate an output frequency that is a multiple of the input frequency.

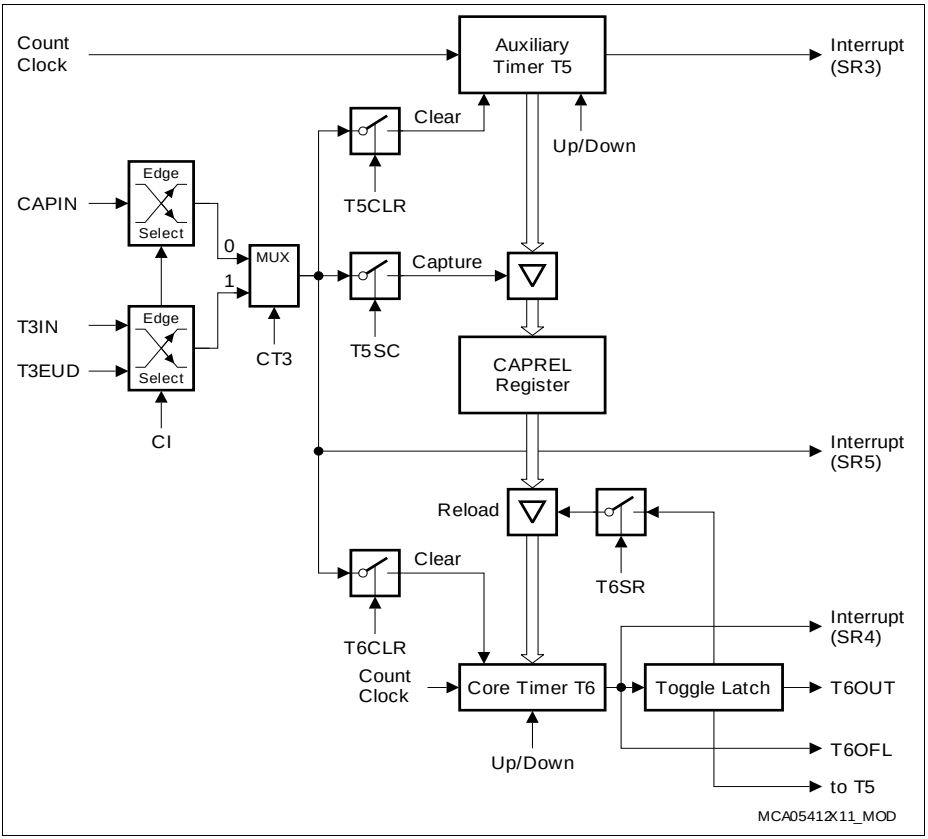


Figure 26-29 Capture/Reload Register CAPREL in Capture-And-Reload Mode

This combined mode can be used to detect consecutive external events which may occur aperiodically, but where a finer resolution, that means, more ‘ticks’ within the time between two external events is required.

For this purpose, the time between the external events is measured using timer T5 and the CAPREL register. Timer T5 runs in Timer Mode counting up with a frequency of e.g. $f_{GPT}/32$. The external events are applied e.g. to pin CAPIN (or other pins supporting

capture mode, see **Table 26-19**). When an external event occurs, the contents of timer T5 are latched into register CAPREL and timer T5 is cleared (T5CLR = 1). Thus, register CAPREL always contains the correct time between two events, measured in timer T5 increments. Timer T6, which runs in Timer Mode counting down with a frequency of e.g. $f_{GPT}/4$, uses the value in register CAPREL to perform a reload on underflow. This means, the value in register CAPREL represents the time between two underflows of timer T6, now measured in timer T6 increments. Since (in this example) timer T6 runs 8 times faster than timer T5, it will underflow 8 times within the time between two external events. Thus, the underflow signal of timer T6 generates 8 'ticks'. Upon each underflow, the interrupt request SR4 will be activated and bit T6OTL will be toggled. The state of T6OTL may be output on pin T6OUT. This signal on T6OUT has 8 times more transitions than the signal which is applied to the selected input (CAPIN in this example).

*Note: The underflow signal of Timer T6 can furthermore be used to clock functional units in other modules, e.g. one or more of the timers of the capture/compare units CCU6 (connections see **Section 26.5.2**). This gives the user the possibility to set compare events based on a finer resolution than that of the external events. This connection is accomplished via signal T6OFL.*

Capture Correction

A certain deviation of the output frequency is generated by the fact that timer T5 will count actual time units (e.g. T5 running at 1 MHz will count up to the value $64_H/100_D$ for a 10 kHz input signal), while T6OTL will only toggle upon an underflow of T6 (i.e. the transition from 0000_H to $FFFF_H$). In the above mentioned example, T6 would count down from 64_H, so the underflow would occur after 101 timing ticks of T6. The actual output frequency then is 79.2 kHz, instead of the expected 80 kHz.

This deviation can be compensated for by using T6 overflows. In this case, T5 counts down and T6 counts up. Upon a signal transition at the selected input pin(s), e.g. on CAPIN, the count value in T5 is captured into CAPREL and T5 is cleared to 0000_H. In its next clock cycle, T5 underflows to $FFFF_H$, and continues to count down with the following clocks. T6 is reloaded from CAPREL upon an overflow, and continues to count up with its following clock cycles (8 times faster in the above example). In this case, T5 and T6 count the same number of steps with their respective internal count frequency.

In the above example, T5 running at 1 MHz will count down to the value $FF9C_H/-100_D$ for a 10 kHz input signal applied e.g. at CAPIN, while T6 counts up from $FF9C_H$ through $FFFF_H$ to 0000_H. So the overflow occurs after 100 timing ticks of T6, and the actual output frequency at T6OUT then is the expected 80 kHz.

However, in this case CAPREL does not directly contain the time between two external events, but rather its 2's complement. Software will have to convert this value, if it is required for the operation.

Combined Capture Modes

For incremental interface applications in particular, several timer features can be combined to obtain dynamic information such as speed, acceleration, or deceleration. The current position itself can be obtained directly from the timer register (T2, T3, T4).

The time information to determine the dynamic parameters is generated by capturing the contents of the free-running timer T5 into register CAPREL. Two trigger sources for this event can be selected:

- Capture trigger on sensor signal transitions
- Capture trigger on position read operations

Capturing on sensor signal transitions is available for timer T3 inputs. This mode is selected by setting bit CT3 and selecting the intended signal(s) via bitfield CI in register T5CON. CAPREL then indicates the time between two selected transitions (measured in T5 counts).

Capturing on position read operations is available for timers T2, T3, and T4. This mode is selected by clearing bit CT3 and selecting the rising edge via bitfield CI in register T5CON. Bitfield ISCAPIN in register PISEL then selects either a read access from T3 or a read access from any of T2 or T3 or T4. CAPREL then indicates the time between two read accesses.

In general, GPT12 has no destructive read mechanisms, except for this special operation mode ‘capture on position read operations’, where register CAPREL is intentionally updated after a read of T3 or T2/T4 (and T5 or T6 are optionally cleared). In this case, also the associated service request flag in register SRC_GPT120CIRQ is set.

Note: If mode ‘capturing on position read operations’ is selected, and the corresponding timer (T3, or any of T2, T3, T4) is read by a debugger, a capture event may be triggered (T5 is captured into CAPREL, and T5 or T6 may optionally be cleared (if T5CLR = 1 or T6CLR = 1)).

These operating modes directly support the measurement of position and rotational speed. Acceleration and deceleration can then be determined by evaluating subsequent speed measurements.

26.2.6 GPT2 Clock Signal Control

All actions within the timer block GPT2 are triggered by transitions of its basic clock. This basic clock is derived from the module clock f_{GPT} by a basic block prescaler, controlled by bitfield BPS2 in register T6CON (see **Figure 26-18**). The count clock can be generated in two different ways:

- **Internal count clock**, derived from GPT2's basic clock via a programmable prescaler, is used for (Gated) Timer Mode.
- **External count clock**, derived from the timer's input pin(s), is used for Counter Mode.

For both ways, the basic clock determines the maximum count frequency and the timer's resolution:

Table 26-12 Basic Clock Selection for Block GPT2

Block Prescaler[1]	BPS2 = 01_B	BPS2 = 00_B[2]	BPS2 = 11_B	BPS2 = 10_B
Prescaling Factor for GPT2: F(BPS2)	F(BPS2) = 2	F(BPS2) = 4	F(BPS2) = 8	F(BPS2) = 16
Maximum External Count Frequency	$f_{GPT}/4$	$f_{GPT}/8$	$f_{GPT}/16$	$f_{GPT}/32$
Input Signal Stable Time	2 x t_{GPT}	4 x t_{GPT}	8 x t_{GPT}	16 x t_{GPT}

1) Please note the non-linear encoding of bitfield BPS2.

2) Default after reset.

Note: The GPT2 module uses a finite state machine to control the actions. Since multiple interactions are possible between the timers (T5, T6) and register CAPREL, these elements are processed sequentially. However, all actions are normally completed within one basic clock cycle. The GPT2 state machine has 4 states (2 states when BPS2 = 01_B) and processes T6 before T5.

Note: When initializing the GPT2 block after reset, and the block prescaler BPS2 in register T6CON needs to be set to a value different from its default value (00_B), it must be initialized first before any mode involving external trigger signals is configured. These modes include counter, capture, and reload mode. Otherwise, unintended count/capture/reload events may occur during the first basic clock cycle.
In this case, or when changing BPS2 during operation of the GPT2 block, disable related interrupts before modification of BPS2, and afterwards clear the corresponding service request flags and re-initialize those registers (T5, T6, CAPREL) that might be affected by a count/capture/reload event.

Internal Count Clock Generation

In Timer Mode and Gated Timer Mode, the count clock for each GPT2 timer is derived from the GPT2 basic clock by a programmable prescaler, controlled by bitfield TxI in the respective timer's control register TxCON.

The count frequency f_{Tx} for a timer Tx and its resolution r_{Tx} are scaled linearly with lower clock frequencies, as can be seen from the following formula:

$$f_{Tx} = \frac{f_{GPT}}{F(BPS2) \times 2^{<TxI>}} \qquad r_{Tx}[\mu s] = \frac{F(BPS2) \times 2^{<TxI>}}{f_{GPT}[MHz]} \tag{26.2}$$

The effective count frequency depends on the common module clock prescaler factor F(BPS2) as well as on the individual input prescaler factor $2^{<TxI>}$. **Table 26-16** summarizes the resulting overall divider factors for a GPT2 timer that result from these cascaded prescalers.

Table 26-13 lists GPT2 timer's parameters (such as count frequency, resolution, and period) resulting from the selected overall prescaler factor and the module clock f_{GPT}. Note that some numbers may be rounded.

Table 26-13 GPT2 Timer Parameters

Module Clock f_{GPT} = 10 MHz			Overall Prescaler Factor	Module Clock f_{GPT} = 40 MHz		
Frequency	Resolution	Period		Frequency	Resolution	Period
5.0 MHz	200 ns	13.11 ms	2	20.0 MHz	50 ns	3.28 ms
2.5 MHz	400 ns	26.21 ms	4	10.0 MHz	100 ns	6.55 ms
1.25 MHz	800 ns	52.43 ms	8	5.0 MHz	200 ns	13.11 ms
625.0 kHz	1.6 µs	104.9 ms	16	2.5 MHz	400 ns	26.21 ms
312.5 kHz	3.2 µs	209.7 ms	32	1.25 MHz	800 ns	52.43 ms
156.25 kHz	6.4 µs	419.4 ms	64	625.0 kHz	1.6 µs	104.9 ms
78.125 kHz	12.8 µs	838.9 ms	128	312.5 kHz	3.2 µs	209.7 ms
39.06 kHz	25.6 µs	1.678 s	256	156.25 kHz	6.4 µs	419.4 ms
19.53 kHz	51.2 µs	3.355 s	512	78.125 kHz	12.8 µs	838.9 ms
9.77 kHz	102.4 µs	6.711 s	1024	39.06 kHz	25.6 µs	1.678 s
4.88 kHz	204.8 µs	13.42 s	2048	19.53 kHz	51.2 µs	3.355 s

External Count Clock Input

The external input signals of the GPT2 block are sampled with the GPT2 basic clock (see **Figure 26-18**). To ensure that a signal is recognized correctly, its current level (high or

low) must be held active for at least one complete sampling period, before changing. A signal transition is recognized if two subsequent samples of the input signal represent different levels. Therefore, a minimum of two basic clock periods are required for the sampling of an external input signal. Thus, the maximum frequency of an input signal must not be higher than half the basic clock.

Table 26-14 summarizes the resulting requirements for external GPT2 input signals.

Table 26-14 GPT2 External Input Signal Limits

GPT2 Basic Clock = 10 MHz		Input Freq. Factor	GPT2 Divider BPS2	Input Phase Duration	GPT2 Basic Clock = 40 MHz	
Max. Input Frequency	**Min. Level Hold Time**				**Max. Input Frequency**	**Min. Level Hold Time**
2.5 MHz	200 ns	$f_{GPT}/4$	01_B	2 x t_{GPT}	10.0 MHz	50 ns
1.25 MHz	400 ns	$f_{GPT}/8$	00_B	4 x t_{GPT}	5.0 MHz	100 ns
625.0 kHz	800 ns	$f_{GPT}/16$	11_B	8 x t_{GPT}	2.5 MHz	200 ns
312.5 kHz	1.6 µs	$f_{GPT}/32$	10_B	16 x t_{GPT}	1.25 MHz	400 ns

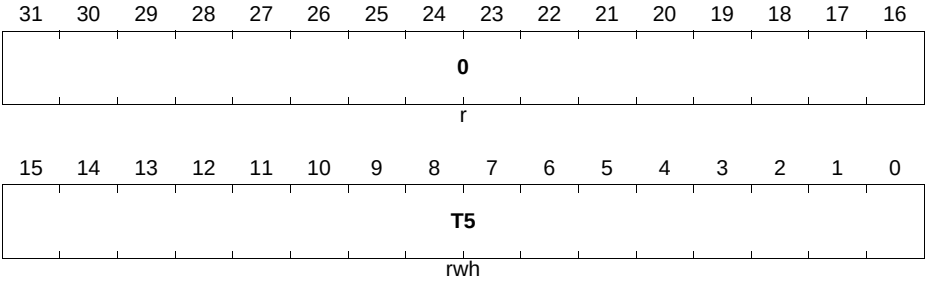
These limitations are valid for all external input signals to GPT2, including the external count signals in Counter Mode and the gate input signals in Gated Timer Mode.

26.2.7 GPT2 Registers

26.2.7.1 GPT2 Timer Registers

T5

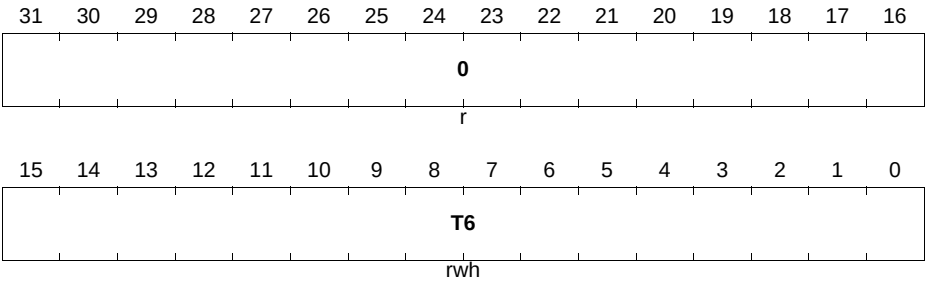
Timer T5 Register **(40_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
T5	[15:0]	rwh	**Timer T5** Contains the current value of Timer T5.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

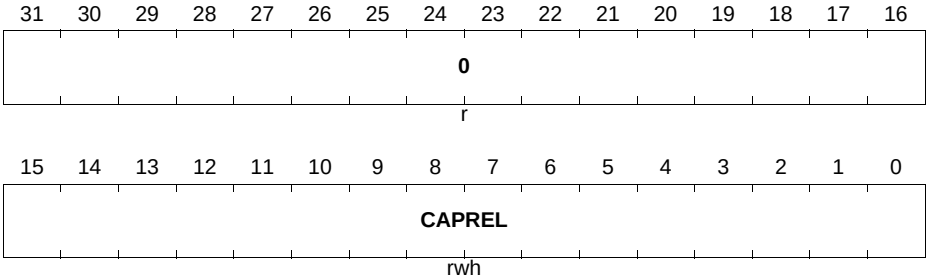
T6

Timer T6 Register **(44_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
T6	[15:0]	rwh	**Timer T6** Contains the current value of Timer T6.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

CAPREL
Capture and Reload Register **(30_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CAPREL	[15:0]	rwh	**Current reload value or Captured value**
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

26.2.7.2 GPT2 Timer Control Registers

GPT2 Core Timer T6 Control Register

T6CON
Timer T6 Control Register **(20_H)** **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T6 SR	T6 CLR	0	BPS2		T6 OTL	T6 OE	T6 UDE	T6 UD	T6 R	T6M			T6I		
rw	rw	r	rw		rwh	rw	rw	rw	rw	rw			rw		

Field	Bits	Type	Description
T6I	[2:0]	rw	**Timer T6 Input Parameter Selection** Depends on the operating mode, see respective sections for encoding: **Table 26-16** for Timer Mode and Gated Timer Mode **Table 26-17** for Counter Mode
T6M	[5:3]	rw	**Timer T6 Mode Control** (Basic Operating Mode) 000_B Timer Mode 001_B Counter Mode 010_B Gated Timer Mode with gate active low 011_B Gated Timer Mode with gate active high 100_B Reserved. Do not use this combination. 101_B Reserved. Do not use this combination. 110_B Reserved. Do not use this combination. 111_B Reserved. Do not use this combination.
T6R	6	rw	**Timer T6 Run Bit** 0_B Timer T6 stops 1_B Timer T6 runs

Field	Bits	Type	Description
T6UD	7	rw	**Timer T6 Up/Down Control**[1)] 0_B Timer T6 counts up 1_B Timer T6 counts down *Note: This bit only controls count direction of T6 if bit T6UDE = 0.*
T6UDE	8	rw	**Timer T6 External Up/Down Enable**[1)] 0_B Count direction is controlled by bit T6UD; input T6EUD is disconnected 1_B Count direction is controlled by input T6EUD
T6OE	9	rw	**Overflow/Underflow Output Enable** 0_B Alternate Output Function Disabled 1_B State of timer T6 toggle latch is output on pin T6OUT
T6OTL	10	rwh	**Timer T6 Overflow Toggle Latch** Toggles on each overflow/underflow of timer T6. Can be set or cleared by software (see separate description)
BPS2	[12:11]	rw	**GPT2 Block Prescaler Control** Selects the basic clock for block GPT2 (see also **Section 26.2.6**) 00_B $f_{GPT}/4$ 01_B $f_{GPT}/2$ 10_B $f_{GPT}/16$ 11_B $f_{GPT}/8$
T6CLR	14	rw	**Timer T6 Clear Enable Bit** 0_B Timer T6 is not cleared on a capture event 1_B Timer T6 is cleared on a capture event
T6SR	15	rw	**Timer T6 Reload Mode Enable** 0_B Reload from register CAPREL Disabled 1_B Reload from register CAPREL Enabled
0	13, [31:16]	r	**Reserved** Read as 0; should be written with 0.

1) See **Table 26-15** for coding of bits T6UD and T6UDE

GPT2 Auxiliary Timer T5 Control Registers

T5CON
Timer T5 Control Register **($1C_H$)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T5 SC	T5 CLR	CI		0	CT3	T5 RC	T5 UDE	T5 UD	T5 R	T5M			T5I		
rw	rwh	rw		rw	rw	rw	rw	rw	rw	rw			rw		

Field	Bits	Type	Description
T5I	[2:0]	rw	**Timer T5 Input Parameter Selection** Depends on the operating mode, see respective sections for encoding: **Table 26-16** for Timer Mode and Gated Timer Mode **Table 26-18** for Counter Mode
T5M	[5:3]	rw	**Timer T5 Mode Control** (Basic Operating Mode) 000_B Timer Mode 001_B Counter Mode 010_B Gated Timer Mode with gate active low 011_B Gated Timer Mode with gate active high 100_B Reserved. Do not use this combination 101_B Reserved. Do not use this combination 110_B Reserved. Do not use this combination 111_B Reserved. Do not use this combination
T5R	6	rw	**Timer T5 Run Bit** 0_B Timer T5 runs 1_B Timer T5 stops *Note: This bit only controls timer T5 if bit T5RC = 0.*
T5UD	7	rw	**Timer T5 Up/Down Control**[1)] 0_B Timer T5 counts up 1_B Timer T5 counts down *Note: This bit only controls count direction of T5 if bit T5UDE = 0.*

Field	Bits	Type	Description
T5UDE	8	rw	**Timer T5 External Up/Down Enable**[1)] 0_B Count direction is controlled by bit T5UD; input T5EUD is disconnected 1_B Count direction is controlled by input T5EUD
T5RC	9	rw	**Timer T5 Remote Control** 0_B Timer T5 is controlled by its own run bit T5R 1_B Timer T5 is controlled by the run bit T6R of core timer T6, not by bit T5R
CT3	10	rw	**Timer T3 Capture Trigger Enable** 0_B Capture trigger from input line CAPIN 1_B Capture trigger from T3 input lines T3IN and/or T3EUD
0	11	rw	**Reserved** Has to be written with 0.
CI	[13:12]	rw	**Register CAPREL Capture Trigger Selection**[2)] 00_B Capture disabled 01_B Positive transition (rising edge) on CAPIN[3)] or any transition on T3IN 10_B Negative transition (falling edge) on CAPIN or any transition on T3EUD 11_B Any transition (rising or falling edge) on CAPIN or any transition on T3IN or T3EUD
T5CLR	14	rw	**Timer T5 Clear Enable Bit** 0_B Timer T5 is not cleared on a capture event 1_B Timer T5 is cleared on a capture event
T5SC	15	rw	**Timer T5 Capture Mode Enable** 0_B Capture into register CAPREL disabled 1_B Capture into register CAPREL enabled
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

1) See **Table 26-15** for encoding of bits T5UD and T5UDE.

2) To define the respective trigger source signal, also bit CT3 must be regarded (see **Table 26-19**).

3) Rising edge must be selected for triggering the capture/clear operation by the internal GPT1 read signals (see bit field ISCAPIN in register **PISEL** and description in section **"Combined Capture Modes" on Page 26-58**).

Encoding of GPT2 Timer Count Direction Control

Table 26-15 GPT2 Timer Count Direction Control

Pin TxEUD	Bit TxUDE	Bit TxUD	Count Direction
X	0	0	Count Up
X	0	1	Count Down
0	1	0	Count Up
1	1	0	Count Down
0	1	1	Count Down
1	1	1	Count Up

Encoding of GPT2 Overall Prescaler Factor in Timer Mode and Gated Timer Mode

Table 26-16 GPT2 Overall Prescaler Factors for Internal Count Clock (Timer Mode and Gated Timer Mode)

Individual Prescaler for Tx	Common Prescaler for Module Clock[1]			
	BPS2 = 01_B	BPS2 = 00_B	BPS2 = 11_B	BPS2 = 10_B
TxI = 000_B	2	4	8	16
TxI = 001_B	4	8	16	32
TxI = 010_B	8	16	32	64
TxI = 011_B	16	32	64	128
TxI = 100_B	32	64	128	256
TxI = 101_B	64	128	256	512
TxI = 110_B	128	256	512	1024
TxI = 111_B	256	512	1024	2048

1) Please note the non-linear encoding of bitfield BPS2.

Encoding of GPT2 Input Edge Selection in Counter Mode

Table 26-17 GPT2 Core Timer T6 (Counter Mode) Input Edge Selection

T6I	Triggering Edge for Counter Increment/Decrement
000_B	None. Counter T6 is disabled
001_B	Positive transition (rising edge) on T6IN
010_B	Negative transition (falling edge) on T6IN
011_B	Any transition (rising or falling edge) on T6IN
$1XX_B$	Reserved. Do not use this combination

Table 26-18 GPT2 Auxiliary Timer T5 (Counter Mode) Input Edge Selection

T5I	Triggering Edge for Counter Increment/Decrement
$X00_B$	None. Counter T5 is disabled
001_B	Positive transition (rising edge) on T5IN
010_B	Negative transition (falling edge) on T5IN
011_B	Any transition (rising or falling edge) on T5IN
101_B	Positive transition (rising edge) of T6 toggle latch T6OTL
110_B	Negative transition (falling edge) of T6 toggle latch T6OTL
111_B	Any transition (rising or falling edge) of T6 toggle latch T6OTL

Encoding of GPT2 Input Selection for Capture and Timer Clear Function

Table 26-19 CAPREL Register Input Edge Selection

CT3	CI	Triggering Signal/Edge for Capture Mode and/or T5/T6 Clear
X	00_B	None. Capture Mode is disabled.
0	01_B	Positive transition (rising edge) on CAPIN, or read operation on selected GPT1 timers[1)].
0	10_B	Negative transition (falling edge) on CAPIN.
0	11_B	Any transition (rising or falling edge) on CAPIN.
1	01_B	Any transition (rising or falling edge) on T3IN.
1	10_B	Any transition (rising or falling edge) on T3EUD.
1	11_B	Any transition (rising or falling edge) on T3IN or T3EUD.

1) Rising edge must be selected for triggering the capture/clear operation by the internal GPT1 read signals (see bit field ISCAPIN in register **PISEL** and description in section **"Combined Capture Modes" on Page 26-58**).

26.3 GPT12 Kernel Register Overview

Table 26-20 summarizes the GPT12 kernel registers and the module external registers and defines their addresses and reset values.

BPI registers are included in a separate Table 26-21.

Table 26-20 Register Overview of GPT12

Register Short Name	Register Long Name	Offset Addr.[1]	Access Mode		Reset	Page Num.
			Read	Write		
PISEL	Port Input Select Register	04_H	U, SV	U, SV, P	Application Reset	26-73
ID	Identification Register	08_H	U, SV	BE	Application Reset	26-75
T2CON	Timer T2 Control Register	10_H	U, SV	U, SV, P	Application Reset	26-32
T3CON	Timer T3 Control Register	14_H	U, SV	U, SV, P	Application Reset	26-30
T4CON	Timer T4 Control Register	18_H	U, SV	U, SV, P	Application Reset	26-34
T5CON	Timer T5 Control Register	1C	U, SV	U, SV, P	Application Reset	26-66
T6CON	Timer T6 Control Register	20_H	U, SV	U, SV, P	Application Reset	26-64
CAPREL	Capture and Reload Register	30_H	U, SV	U, SV, P	Application Reset	26-63
T2	Timer T2 Register	34_H	U, SV	U, SV, P	Application Reset	26-28
T3	Timer T3 Register	38_H	U, SV	U, SV, P	Application Reset	26-28
T4	Timer T4 Register	$3C_H$	U, SV	U, SV, P	Application Reset	26-29
T5	Timer T5 Register	40_H	U, SV	U, SV, P	Application Reset	26-62
T6	Timer T6 Register	44_H	U, SV	U, SV, P	Application Reset	26-62

1) The absolute register address is calculated as follows:
Module Base Address + Offset Address (shown in this column)

26.4 General Module Operation

This section provides information about the:

- Input Selection (see **Section 26.4.1**)
- OCDS Suspend Functionality (see **Section 26.4.2**)
- Miscellaneous GPT12 Register Description (see **Section 26.4.3**)
- BPI Register Description (see **Section 26.4.4**)

26.4.1 Input Selection

Each GPT12 input signal can be selected from a vector of two or four possible inputs by programming the port input select register **PISEL**. This permits to adapt the pin functionality of the device to the application requirements.
The output pins for the module output signals are chosen in the ports.

Naming convention (example):
The input vector T3IN[D:A] for input signal T3IN is composed of the signals T3INA to T3IND.

*Note: All functional inputs of the GPT12 module are synchronized to the internal basic clock of the GPT1 and GPT2 block, respectively. An edge of an input signal can only be correctly recognized if the high phase and the low phase are longer than one basic clock period. See **Table 26-5** for GPT1 external input signal limits, and **Table 26-14** for GPT2 external input signal limits.*

26.4.2 OCDS Suspend

The behavior of GPT12 upon an OCDS suspend request is controlled by the **OCS** register. GPT12 supports only Hard Suspend Mode.

Hard Suspend Mode

In Hard Suspend Mode the GPT12 kernel clock is switched off immediately. Reading and writing of registers is possible but will enable the kernel clock for a few cycles.

Attention: Register accesses with clocking in Hard Suspend Mode can have unintended side effects like signals becoming and staying active. This can affect also other modules, so a GPT12 kernel reset might not be sufficient to bring the system into a defined state.

26.4.3 Miscellaneous GPT12 Registers

26.4.3.1 Port Input Select Register

Register PISEL contains bit fields selecting the actual input signal for the module inputs.

PISEL
Port Input Select Register **(04_H)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISCAPIN		IST6 EUD	IST6 IN	IST5 EUD	IST5 IN	IST4EUD		IST4IN		IST3EUD		IST3IN		IST2 EUD	IST2 IN
rw		rw	rw	rw	rw	rw		rw		rw		rw		rw	rw

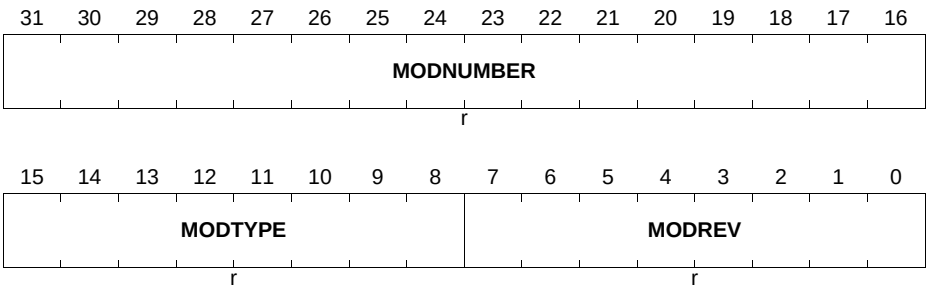
Field	Bits	Type	Description
IST2IN	0	rw	**Input Select for T2IN** 0_B Signal T2INA is selected 1_B Signal T2INB is selected
IST2EUD	1	rw	**Input Select for T2EUD** 0_B Signal T2EUDA is selected 1_B Signal T2EUDB is selected
IST3IN	[3:2]	rw	**Input Select for T3IN** 00_B Signal T3INA is selected 01_B Signal T3INB is selected 10_B Signal T3INC is selected 11_B Signal T3IND is selected
IST3EUD	[5:4]	rw	**Input Select for T3EUD** 00_B Signal T3EUDA is selected 01_B Signal T3EUDB is selected 10_B Signal T3EUDC is selected 11_B Signal T3EUDD is selected

Field	Bits	Type	Description
IST4IN	[7:6]	rw	**Input Select for T4IN** 00_B Signal T4INA is selected 01_B Signal T4INB is selected 10_B Signal T4INC is selected 11_B Signal T4IND is selected
IST4EUD	[9:8]	rw	**Input Select for T4EUD** 00_B Signal T4EUDA is selected 01_B Signal T4EUDB is selected 10_B Signal T4EUDC is selected 11_B Signal T4EUDD is selected
IST5IN	10	rw	**Input Select for T5IN** 0_B Signal T5INA is selected 1_B Signal T5INB is selected
IST5EUD	11	rw	**Input Select for T5EUD** 0_B Signal T5EUDA is selected 1_B Signal T5EUDB is selected
IST6IN	12	rw	**Input Select for T6IN** 0_B Signal T6INA is selected 1_B Signal T6INB is selected
IST6EUD	13	rw	**Input Select for T6EUD** 0_B Signal T6EUDA is selected 1_B Signal T6EUDB is selected
ISCAPIN	[15:14]	rw	**Input Select for CAPIN** 00_B Signal CAPINA is selected 01_B Signal CAPINB is selected 10_B Signal CAPINC (Read trigger from T3) is selected 11_B Signal CAPIND (Read trigger from T2 or T3 or T4) is selected
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

26.4.3.2 Identification Register

The GPT12 Module Identification Register ID contains read-only information about the module identification number and its revision.

ID
Identification Register **(08_H)** **Reset Value: 0068 C0XX$_H$**



Field	Bits	Type	Description
MODREV	[7:0]	r	**Module Revision Number** This bit field indicates the revision number of the TC21x/TC22x/TC23x module (01_H = first revision).
MODTYPE	[15:8]	r	**Module Type** This bit field is $C0_H$. It defines a 32-bit module
MODNUMBER	[31:16]	r	**Module Number** This bit field defines the module identification number. For the GPT12 module the module identification number is 68_H.

26.4.4 BPI Registers

This section describes the registers of the BPI (Bus Peripheral Interface). **Figure 26-30** shows all registers associated with the BPI of a GPT12 module.

BPI Registers Overview

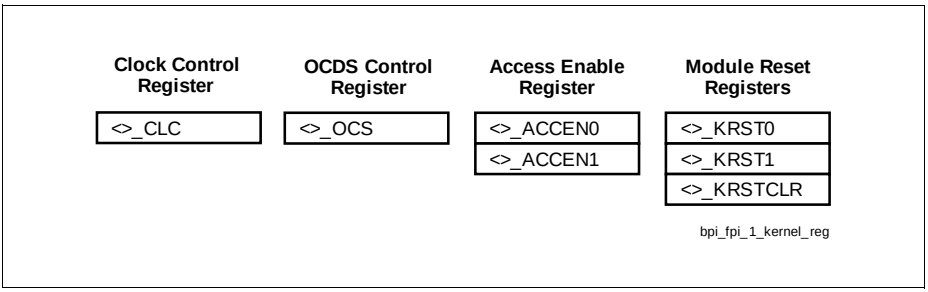


Figure 26-30 BPI Registers

Table 26-21 gives an overview of the BPI registers, including their access modes and reset class.

Table 26-21 Registers Overview - BPI Registers

Register Short Name	Description	Offset Addr.	Access Mode		Reset	Page Num.
			Read	Write		
CLC	Clock Control Register	00_H	U, SV	SV, E, P	Application Reset	26-78
-	Kernel Registers	-	-	-	-	-
OCS	OCDS Control and Status Register	$E8_H$	U, SV	SV, P	Debug Reset	26-79
KRSTCLR	Reset Status Clear Register	EC_H	U, SV	SV, E, P	Application Reset	26-84
KRST1	Reset Control Register 1	$F0_H$	U, SV	SV, E, P	Application Reset	26-83
KRST0	Reset Control Register 0	$F4_H$	U, SV	SV, E, P	Application Reset	26-82
ACCEN1	Access Enable Register 1	$F8_H$	U, SV	SV, SE	Application Reset	26-81
ACCEN0	Access Enable Register 0	FC_H	U, SV	SV, SE	Application Reset	26-80

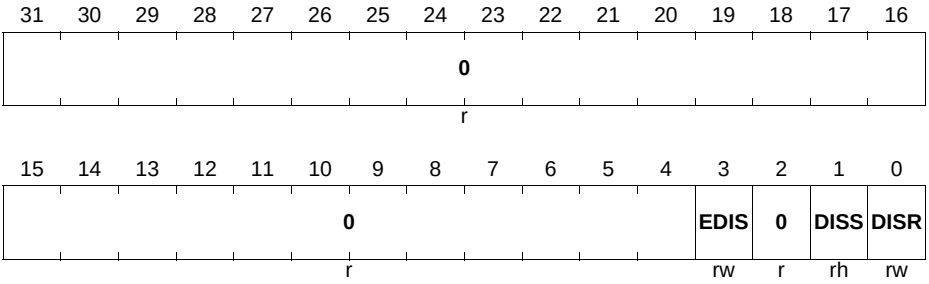
Note: Register bits marked “r” in the following register description are virtual registers and do not contain flip-flops. They are always read as 0.

26.4.4.1 System Registers

Clock Control Register (CLC)

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI for the GPT12 module. Where a module kernel is connected to the CLC clock control interface, CLC controls the f_{GPT} module clock signal and Sleep Mode for the module.

CLC
Clock Control Register (00_H) Reset Value: $0000\ 0003_H$



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. 0_B Module disable is not requested. 1_B Module disable is requested.
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module. 0_B Module is enabled. 1_B Module is disabled.
EDIS	3	rw	**Sleep Mode Enable Control** Used to control module's sleep mode. 0_B Sleep Mode request is regarded. Module is enabled to go into Sleep Mode. 1_B Sleep Mode request is disregarded: Sleep Mode cannot be entered upon a request.

Field	Bits	Type	Description
0	[31:16], [15:4], 2	r	**Reserved** Read as 0; should be written with 0.

Note: Upon an accepted Sleep Mode request (with EDIS = '1'), or upon a disable request (DISR = '1'), the GPT12 kernel clock is switched off immediately. Therefore, software should ensure that the system controlled by the GPT12 kernel has reached a safe state before triggering a Sleep Mode or module disable request.

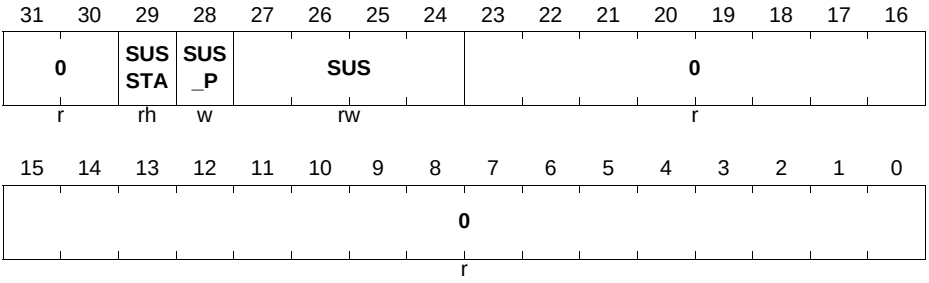
OCDS Control and Status Register (OCS)

The OCDS Control and Status (OCS) register is cleared by Debug Reset.

The OCS register can only be written when the OCDS is enabled. If OCDS is being disabled, the OCS register value will not change. When OCDS is disabled the OCS suspend control is ineffective.

Write access is 32 bit wide only and requires Supervisor Mode.

OCS
OCDS Control and Status Register ($E8_H$) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
SUS	[27:24]	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0_H Will not suspend 1_H Hard suspend. Clock is switched off immediately. **others**, reserved
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.

Field	Bits	Type	Description
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	[23:0], [31:30]	r	**Reserved** Read as 0; must be written with 0.

Access Enable Register 0 (ACCEN0)

The Access Enable Register 0 controls write access[1)] for transactions with the on chip bus master TAG ID 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000_B, EN1 -> TAG ID 000001_B, ... , EN31 -> TAG ID 011111_B.

ACCEN0
Access Enable Register 0 **(FC_H)** **Reset Value: $FFFF\ FFFF_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn **(n = 0-31)**	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

1) The BPI Access Enable functionality controls only write transactions to the CLC, OCS, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers.

Access Enable Register 1 (ACCEN1)

The Access Enable Register 1 controls write access[1)] for transactions with the on chip bus master TAG ID 100000_B to 111111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI is prepared for a 6-bit TAG ID. ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the AURIX devices.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000_B, EN1 -> TAG ID 100001_B, ... , EN31 -> TAG ID 111111_B.

ACCEN1
Access Enable Register 1 **($F8_H$)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
0	[31:0]	r	**Reserved** Read as 0; should be written with 0.

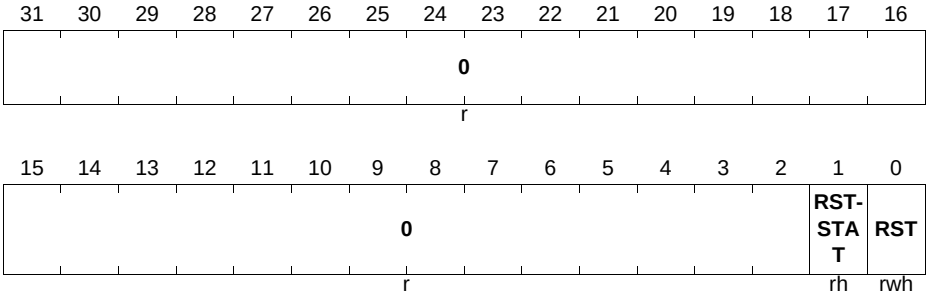
Kernel Reset Register 0 (KRST0)

The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers. The RST bit will be re-set (cleared to ‘0’) by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI in the same clock cycle the RST bit is re-set by the BPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the KRSTCLR.CLR register bit.

Note: During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

KRST0
Kernel Reset Register 0 **($F4_H$)** **Reset Value: 0000 0000_H**

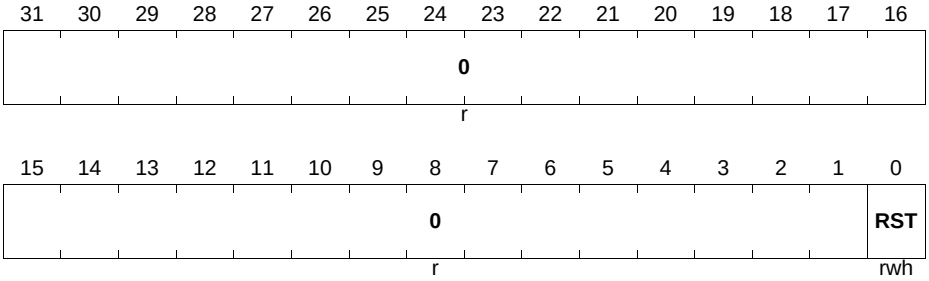


Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) by the BPI after the kernel reset was executed.
RSTSTAT	1	rw	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI after the execution of a kernel reset in the same clock cycle both reset bits. 0_B No kernel reset was executed 1_B Kernel reset was executed This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register.
0	[31:2]	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1 (KRST1)

The Kernel Reset Register 1 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers. The RST bit will be re-set (cleared to ‘0’) by the BPI with the end of the BPI kernel reset sequence.

KRST1
Kernel Reset Register 1 **($F0_H$)** **Reset Value: 0000 0000_H**



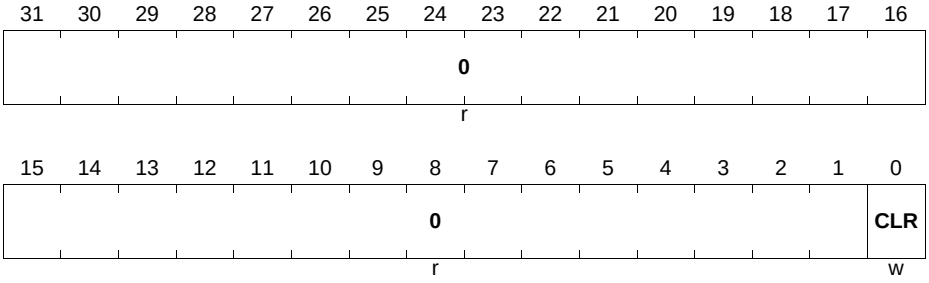
Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) by the BPI after the kernel reset was executed.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Status Clear Register (KRSTCLR)

The Kernel Reset Status Clear Register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR

Kernel Reset Status Clear Register (EC_H) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT Read always as 0.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

26.5 Implementation of the GPT12 Module

This chapter describes the implementation of the GPT12 module in the TC21x/TC22x/TC23x device.

26.5.1 Address Map

There is one GPT12 kernel implemented in the TC21x/TC22x/TC23x, namely GPT120.

Table 26-22 Registers Address Space

Module	Base Address	End Address	Note
GPT120	F000 2E00$_H$	F000 2EFF$_H$	

26.5.2 Module Connections

The following table shows the digital connections of the GPT12 module with other modules or pins in the TC21x/TC22x/TC23x device.

The GPT12 module is clocked with the SPB_Bus clock, so $f_{GPT} = f_{SPB}$.

Note: Signals that are not available in the 100- and 80-pin package variant are specifically marked, see footnote [1].

Note: Signals that are not available in the 80-pin package variant are specifically marked, see footnote [2].

Table 26-23 GPT120 Digital Connections in TC21x/TC22x/TC23x

Signal	from/to Module	I/O to GPT120	Can be used to/as
T2INA	P00.7[1)]	I	count input signals for timer T2
T2INB	P33.7	I	
T2EUDA	P00.8[1)]	I	direction input signals for timer T2
T2EUDB	P33.6	I	
T3INA	P02.6	I	count input signals for timer T3
T3INB	P13.1[2)]	I	
T3INC	ERU_PDOUT4	I	
T3IND	0	I	
T3EUDA	P02.7	I	direction input signals for timer T3
T3EUDB	P13.2[2)]	I	
T3EUDC	0	I	
T3EUDD	0	I	

Table 26-23 GPT120 Digital Connections in TC21x/TC22x/TC23x (cont'd)

Signal	from/to Module	I/O to GPT120	Can be used to/as
T3OUT	P10.6 P21.6 ERU_IN42	O	count output signal for timer T3
T4INA	P02.8	I	count input signals for timer T4
T4INB	P13.3[2)]	I	
T4INC	0	I	
T4IND	0	I	
T4EUDA	P00.9[1)]	I	direction input signals for timer T4
T4EUDB	P33.5	I	
T4EUDC	0	I	
T4EUDD	0	I	
T5INA	P21.7	I	count input signals for timer T5
T5INB	P10.3[1)]	I	
T5EUDA	P21.6	I	direction input signals for timer T5
T5EUDB	P10.1[1)]	I	
T6INA	P20.3[1)]	I	count input signals for timer T6
T6INB	P10.2[1)]	I	
T6EUDA	P20.0[1)]	I	direction input signals for timer T6
T6EUDB	P13.0[2)]	I	
T6OUT	P10.5 P21.7 ERU_IN52	O	count output signal for timer T6
T6OFL	CCU60_T12HRF CCU60_T13HRF CCU61_T12HRF CCU61_T13HRF	O	over/under-flow signal from timer T6
CAPINA	P13.2[2)]	I	capture/timer clear trigger input signals
CAPINB	ERU_PDOUT6	I	
SR0	SRC_GPT120T2	O	GPT120 Timer 2 Service Request
SR1	SRC_GPT120T3	O	GPT120 Timer 3 Service Request
SR2	SRC_GPT120T4	O	GPT120 Timer 4 Service Request

Table 26-23 GPT120 Digital Connections in TC21x/TC22x/TC23x (cont'd)

Signal	from/to Module	I/O to GPT120	Can be used to/as
SR3	SRC_GPT120T5	O	GPT120 Timer 5 Service Request
SR4	SRC_GPT120T6	O	GPT120 Timer 6 Service Request
SR5	SRC_GPT120CIRQ	O	GPT120 CAPREL Service Request

1) not available for devices in 100- and 80-pin package

2) not available for devices in 80-pin package

27 Versatile Analog-to-Digital Converter (VADC)

The TC21x/TC22x/TC23x provides a series of analog input channels connected to a cluster of Analog/Digital Converters using the Successive Approximation Register (SAR) principle to convert analog input values (voltages) to discrete digital values.
The TC21x/TC22x/TC23x is based on individual SAR converters with dedicated Sample&Hold units.
The number of analog input channels and ADCs depends on the chosen product type (please refer to **"Product-Specific Configuration" on Page 27-167**).

Each converter of the ADC cluster can operate independent of the others, controlled by a dedicated set of registers and triggered by a dedicated group request source. The results of each channel can be stored in a dedicated channel-specific result register or in a group-specific result register.

A background request source can access all analog input channels that are not assigned to any group request source. These conversions are executed with low priority. The background request source can, therefore, be regarded as an additional background converter.

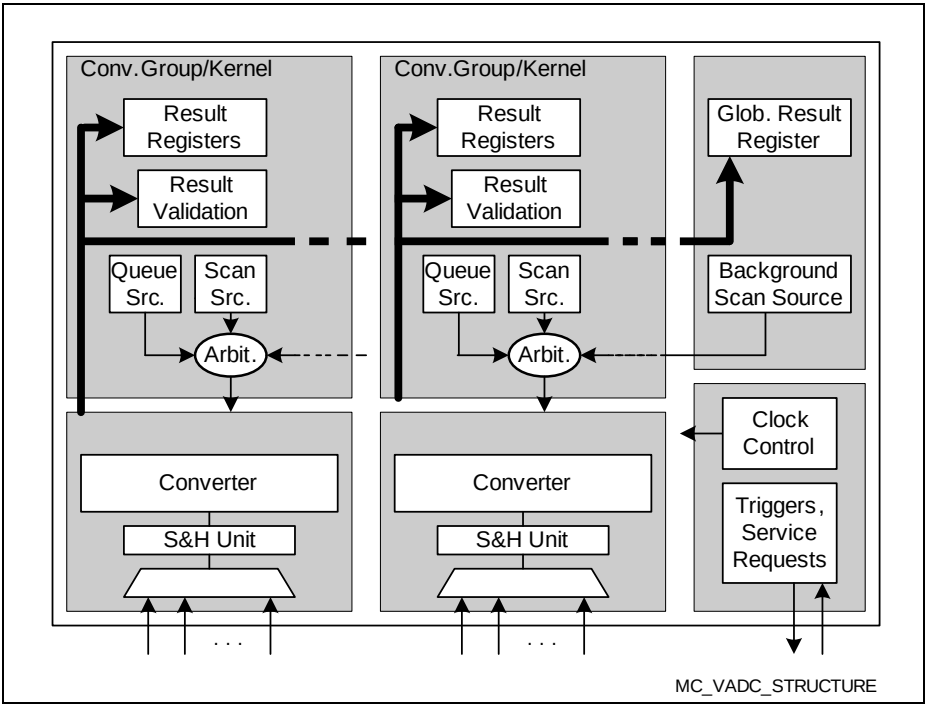


Figure 27-1 ADC Structure Overview

You will find the following major sections within this chapter:

- **"Introduction and Basic Structure" on Page 27-4**
- **"Configuration of General Functions" on Page 27-14**
- **"Analog Converter Control" on Page 27-30**
- **"Conversion Request Generation" on Page 27-32**
  - **"Queued Request Source Handling" on Page 27-34**
  - **"Channel Scan Request Source Handling" on Page 27-64**
- **"Request Source Arbitration" on Page 27-80**
- **"Analog Input Channel Configuration" on Page 27-90**
- **"Conversion Result Handling" on Page 27-110**
- **"Synchronization of Conversions" on Page 27-132**
- **"Safety Features" on Page 27-139**
- **"External Multiplexer Control" on Page 27-148**
- **"Service Request Generation" on Page 27-153**
- **"Implementation into the TC21x/TC22x/TC23x" on Page 27-167** including a **"Summary of Registers and Locations" on Page 27-169**
- **"Use Case Example for VADC" on Page 27-182**

Feature List

The following features describe the functionality of the ADC cluster:

- Nominal analog supply voltage 5.0 V, operation at 3.3 V with reduced performance
- Input voltage range from 0 V up to analog supply voltage
- Standard (V_{AREF}) and alternate (CH0) reference voltage source selectable for each channel to support ratiometric measurements and different signal scales
- 2 independent converters with up to 12+2 analog input channels (4 converters in the ADAS device, see **"Product-Specific Configuration" on Page 27-167**)
- External analog multiplexer control, including adjusted sample time and scan support
- Conversion speed and sample time adjustable to adapt to sensors and reference
- Conversion time below 1 µs (depending on result width and sample time)
- Flexible source selection and arbitration
  - Programmable arbitrary conversion sequence (single or repeated)
  - Configurable auto scan conversion (single or repeated) on each converter
  - Configurable auto scan conversion (single or repeated) in the background (all converters)
  - Conversions triggered by software, timer events, or external events
  - Cancel-inject-restart mode for reduced conversion delay on priority channels
- Powerful result handling
  - Selectable result width of 8/10/12 bits
  - Fast Compare Mode
  - Independent result registers
  - Configurable limit checking against programmable border values
  - Data rate reduction through adding a selectable number of conversion results

  - FIR/IIR filter with selectable coefficients
- Flexible service request generation based on selectable events
- Built-in safety features
  - Configurable register access protection supports safety applications
  - Broken wire detection with programmable default levels
  - Multiplexer test mode to verify signal path integrity
- Support of suspend and power saving modes

Table 27-1 Abbreviations used in ADC chapter

Abbreviation	Meaning
ADAS	Advanced Driver Assistance System
ADC	Analog to Digital Converter
DMA	Direct Memory Access (controller)
DNL	Differential Non-Linearity (error)
INL	Integral Non-Linearity (error)
LSB_n	Least Significant Bit: finest granularity of the analog value in digital format, represented by one least significant bit of the conversion result with n bits resolution (measurement range divided in 2^n equally distributed steps)
SCU	System Control Unit of the device
TUE	Total Unadjusted Error

27.1 Introduction and Basic Structure

The Versatile Analog to Digital Converter module (VADC) of the TC21x/TC22x/TC23x comprises a set of converter blocks that can be operated either independently or via a common request source that emulates a background converter. Each converter block is equipped with a dedicated input multiplexer and dedicated request sources, which together build separate groups, each assigned to a kernel.

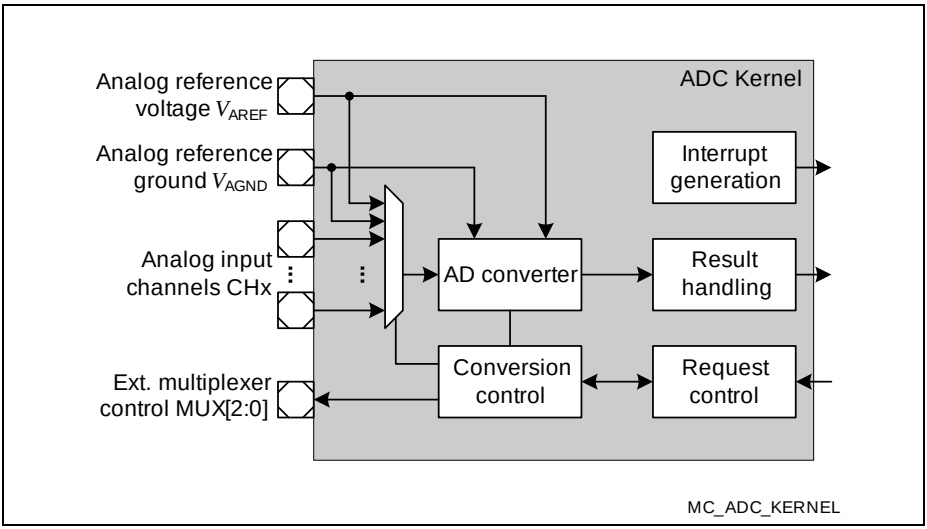


Figure 27-2 ADC Kernel Block Diagram

This basic structure supports application-oriented programming and operating while still providing general access to all resources. The almost identical converter groups allow a flexible assignment of functions to channels.

A set of functional units (described further down in this section) can be configured according to the requirements of a given application. These units build a path from the input signals to the digital results.

Each kernel provides a dedicated Sample&Hold unit connected to the input multiplexer and the converter.

The basic module clock f_{ADC} is connected to the system clock signal f_{SPB}.

Conversion Modes and Request Sources

Analog/Digital conversions can be requested by several request sources (2 group request sources and the background request source) and can be executed in several conversion modes. The request sources can be enabled concurrently with configurable priorities.

- **Fixed Channel Conversion (single or continuous)**
  A specific channel source requests conversions of one selectable channel (once or repeatedly)
- **Auto Scan Conversion (single or continuous)**
  A channel scan source (request source 1 or 2) requests auto scan conversions of a configurable linear sequence of all available channels (once or repeatedly)
- **Channel Sequence Conversion (single or continuous)**
  A queued source (request source 0 or 3) requests a sequence of conversions of up to 8 arbitrarily selectable channels (once or repeatedly)

The conversion modes can be used concurrently by the available request sources, i.e. conversions in different modes can be enabled at the same time. Each source can be enabled separately and can be triggered by external events, such as edges of PWM or timer signals, or pin transitions.

Request Source Control

Because all request sources can be enabled at the same time, an arbiter resolves concurrent conversion requests from different sources. Each source can be triggered by external signals, by on-chip signals, or by software.

Requests with higher priority can either cancel a running lower-priority conversion (cancel-inject-repeat mode) or be converted immediately after the currently running conversion (wait-for-start mode). If the target result register has not been read, a conversion can be deferred (wait-for-read mode).

Certain channels can also be synchronized with other ADC kernels, so several signals can be converted in parallel.

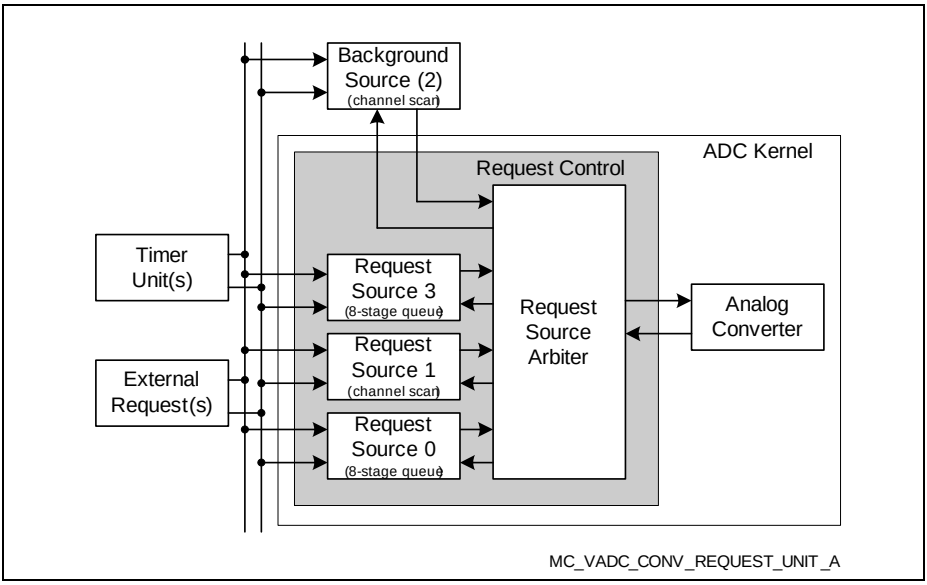


Figure 27-3 Conversion Request Unit

Input Channel Selection

The analog input multiplexer selects one of the available analog inputs (CH0 - CHx[1)]) to be converted. Three sources can select a linear sequence, an arbitrary sequence, or a specific channel. The priorities of these sources can be configured.

Additional external analog multiplexers can be controlled automatically, if more separate input channels are required than are built in (see **Section 27.12**).

*Note: Not all analog input channels are necessarily available in all packages, due to pin limitations. Please refer to the implementation description in **Section 27.14**.*

Conversion Control

Conversion parameters, such as sample phase duration or result resolution can be configured for 4 input classes (2 group-specific classes, 2 global classes). Each channel can be individually assigned to one of these input classes.

Each channel can select either the standard reference voltage or the alternate reference voltage (restrictions see **"Product-Specific Configuration" on Page 27-167**).

1) The availablity of input channels depends on the package of the used product type. A summary can be found in **Section 27.14.3**.

The input channels can, thus, be adjusted to the type of sensor (or other analog sources) connected to the ADC.

This unit also controls the built-in multiplexer and external analog multiplexers, if selected.

Analog/Digital Converter

The selected input channel is converted to a digital value by first sampling the voltage on the selected input and then generating the selected number of result bits.

For 12-bit conversions, post-calibration is executed after converting the channel.

For broken wire detection (see **Section 27.11.1**), the converter network can be preloaded before sampling the selected input channel.

Result Handling

The conversion results of each analog input channel can be directed to one of 16 group-specific result registers and one global result register to be stored there. A result register can be used by a group of channels or by a single channel.

The wait-for-read mode avoids data loss due to result overwrite by blocking a conversion until the previous result has been read.

Data reduction (e.g. for digital anti-aliasing filtering) can automatically add up to 4 conversion results before issuing a service request.

Alternatively, an FIR or IIR filter can be enabled that preprocesses the conversion results before sending them to the result register.

Also, result registers can be concatenated to build FIFO structures that store a number of conversion results without overwriting previous data. This increases the allowed CPU latency for retrieving conversion data from the ADC.

Service Request Generation

Several ADC events can issue service requests to CPU or DMA:

- **Source events** indicate the completion of a conversion sequence in the corresponding request source. This event can be used to trigger the setup of a new sequence.
- **Channel events** indicate the completion of a conversion for a certain channel. This can be combined with limit checking, so interrupt are generated only if the result is within a defined range of values.
- **Result events** indicate the availability of new result data in the corresponding result register. If data reduction mode is active, events are generated only after a complete accumulation sequence.

Each event can be assigned to one of eight service request nodes. This allows grouping the requests according to the requirements of the application.

Safety Features

Safety-aware applications are supported with mechanisms that help to ensure the integrity of a signal path.

Broken-wire-detection (BWD) preloads the converter network with a selectable level before sampling the input channel. The result will then reflect the preload value if the input signal is no more connected. If buffer capacitors are used, a certain number of conversions may be required to reach the failure indication level.

Pull Down Diagnostics (PDD) connects an additional strong pull-down device to an input channel. A subsequent conversion can then confirm the expected modified signal level. This allows to check the proper connection of a signal source (sensor) to the multiplexer.

Multiplexer Diagnostics (MD) connects a weak pull-up or pull-down device to an input channel. A subsequent conversion can then confirm the expected modified signal level. This allows to check the proper operation of the multiplexer.

Converter Diagnostics (CD) connects an alternate signal to the converter. A subsequent conversion can then confirm the proper operation of the converter.

Automatic Test Conversions can be scheduled using request source 3. This queued request source can execute an arbitrary test sequence. Please refer to **Section 27.11.4**.

27.2 Electrical Models

Each conversion of an analog input voltage to a digital value consists of two consecutive phases:

- During the sample phase, the input voltage is sampled and stored.
  The **Input Signal Path** is a simplified model for this.
- During the conversion phase the stored voltage is converted to a digital result.
  The **Reference Voltage Path** is a simplified model for this.

Input Signal Path

The ADC of the TC21x/TC22x/TC23x uses a switched capacitor field represented by C_{AINSW} (small parasitic capacitances are present at each input pin). During the sample phase, the capacitor field C_{AINSW} is connected to the selected analog input CHx via the input multiplexer (modeled by ideal switches and series resistors R_{AIN}).

The switch to CHx is closed during the sample phase and connects the capacitor field to the input voltage V_{AINx}.

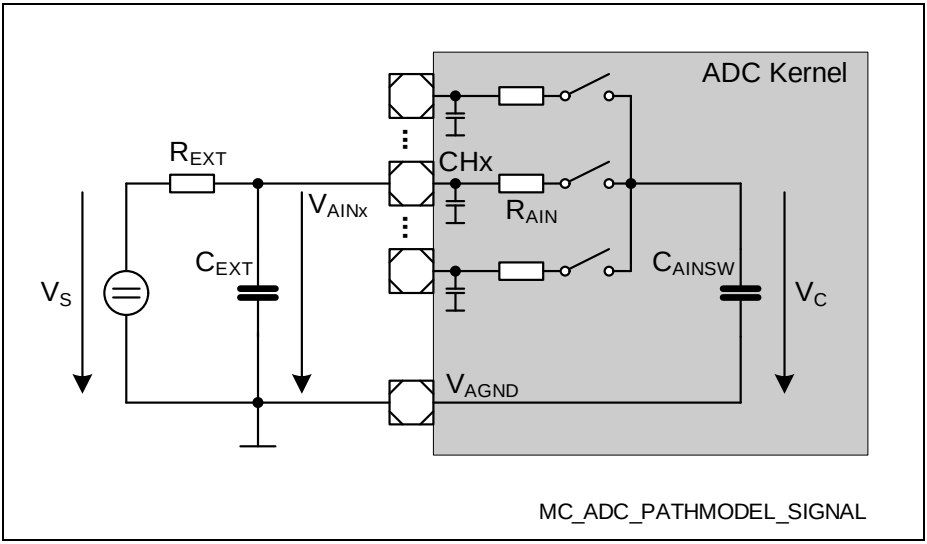


Figure 27-4 Signal Path Model

A simplified model for the analog input signal path is given in **Figure 27-4**. An analog voltage source (value V_S) with an internal impedance of R_{EXT} delivers the analog input that should be converted.

During the sample phase the corresponding switch is closed and the capacitor field C_{AINSW} is charged. Due to the low-pass behavior of the resulting RC combination, the

voltage V_C to be actually converted does not immediately follow V_S. The value R_{EXT} of the analog voltage source and the desired precision of the conversion strongly define the required length of the sample phase.

To reduce the influence of R_{EXT} and to filter input noise, it is recommended to introduce a fast external blocking capacitor C_{EXT} at the analog input pin of the ADC. Like this, mainly C_{EXT} delivers the charge during the sample phase. This structure allows a significantly shorter sample phase than without a blocking capacitor, because the low-pass time constant defining the sample time is mainly given by the values of R_{AIN} and C_{AINSW}.

The resulting low-pass filter of R_{EXT} (usually a parameter of the signal source) and C_{EXT} should be dimensioned according to the application's requirements:

- For quickly changing dynamic signals, a smaller capacitor allows V_{AINx} to follow V_S between two sample phases of the same analog input channel.
- For high-precision conversions, an external blocking capacitor C_{EXT} in the range of at least $2^n \times C_{AINSW}$ keeps the voltage change of V_{AINx} during the sample phase below 1 LSB_n. This voltage change is due to the charge redistribution between C_{EXT} and C_{AINSW}.

Leakage current through the analog input structure of the ADC can generate a voltage drop over R_{EXT}, introducing an error. The ADC input leakage current increases at high temperature and if the input voltage level is close to the analog supply ground V_{SS} or to the analog power supply V_{DDPA}. The input leakage current of an ADC channel can be reduced by avoiding input voltages close to the supplies.

An overload condition (input voltage exceeds the supply range) at adjacent analog inputs injects an additional leakage current (defined by a coupling factor) .

The capacitor C_{AINSW} is automatically precharged to a voltage of approximately half the standard reference voltage V_{AREF} to minimize the average difference between V_{AINx} and V_C at the beginning of a sample phase. Due to varying parameters and parasitic effects, the precharge voltage of C_{AINSW} is typically smaller than V_{AREF} / 2.

Reference Voltage Path

During the conversion phase, parts of the capacitor field (represented by C_{AREFSW}) are switched to a reference input (V_{AREF} or CH0) or to V_{AGND}. Using CH0 as alternate reference source allows conversions of 5.0 V and 3.3 V based analog input signals with the same ADC kernel.

Stable and noise-free reference and analog supply voltages support accurate conversion results. Because noise can also be introduced from other modules (e.g. switching pins), it is strongly recommended to carefully decouple analog from digital signal domains.

The switching of parts of C_{AREFSW} requires a dynamic current at the selected reference input. The impedance R_{REFEXT} of the reference voltage source V_R has to be low enough to supply the reference current during the conversion phase. An external blocking capacitor C_{REFEXT} can supply the peak currents and minimize the current to be delivered by the reference source.

The reference current I_{AREF} introduces a voltage drop at R_{REFEXT} that should not be neglected for the calculation of the overall accuracy. The average reference current during a conversion depends on the reference voltage level and the time t_{CONV} between two conversion starts: $I_{AREF} = C_{AREFSW} \times V_{AREF} / t_{CONV}$.

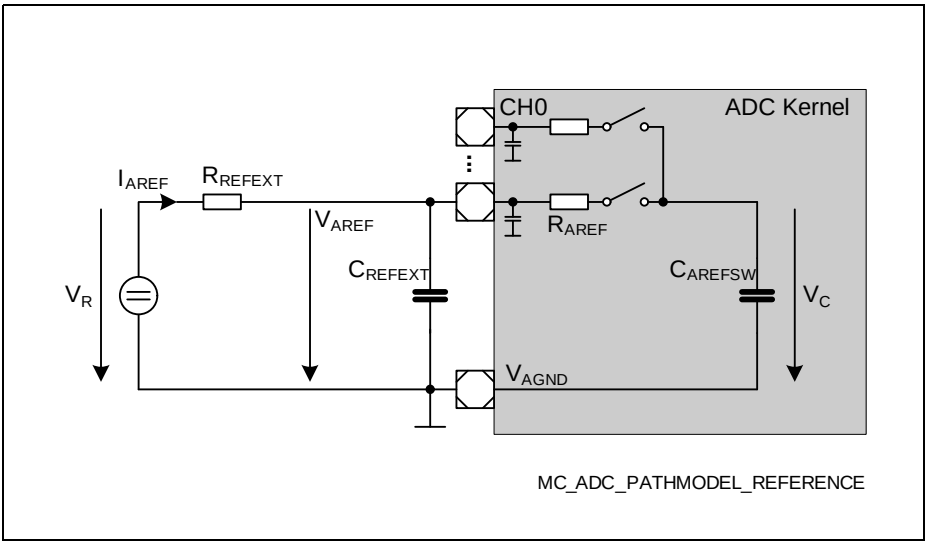


Figure 27-5 Reference Path Model

Reference Signal Connection

The reference voltages (V_{AREF}, V_{AGND}) have a strong influence on the digital result of a conversion. It is, therefore, recommended to carefully avoid noise of these inputs. The recommended filter structures (see figure) help to cancel high-frequency noise.

- Capacitors (C_{REFEXT}) between corresponding reference pins (V_{AREF}, V_{AGND}) provide peak currents for the conversion steps[1)]
- Capacitors (C_N) between reference ground and signal ground (V_{AGND}, V_{SS}) and the inductance of the connections attenuate noise
- Use low-ESR capacitors connected closely to the pins
- Connect reference pins via separate lines to star points close to the sensor supply
- Additional resistors in the reference lines (R_F) must be able to carry the average reference current, to avoid inaccuracy due to voltage drop
- The reference ground lines are connected to the signal ground in many applications

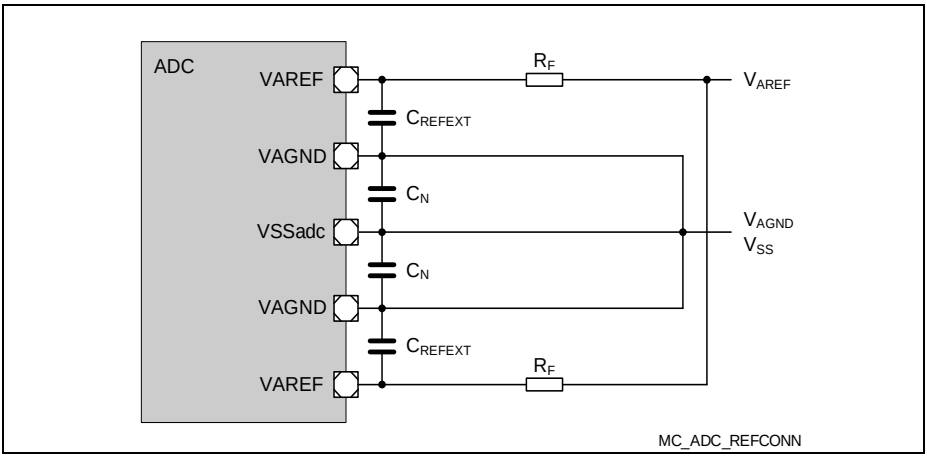


Figure 27-6 Connection of Reference Signals

Due to the charge redistribution between C_{REFEXT} and C_{AREFSW}, the voltage V_{AREF} decreases during the conversion phase. The error introduced by this effect is limited by the external blocking capacitor. $C_{REFEXT} \geq 2^n \times C_{AREFSW}$ limits this error to 1/2 LSB_n (double C_{REFEXT} to limit the error to 1/4 LSB_n).

Assuming 12-bit conversions and C_{AREFSW} = 30 pF leads to:
$C_{REFEXT} > 2^{12} \times 30$ pF = 123 nF, 100 nF are a good approximation.

The noise filtering capacitor C_N depends on the board properties (e.g. 100 nF).

1) The required capacitor value here depends on the resolution and accuracy.

Transfer Characteristics and Error Definitions

The transfer characteristic of the ADC describes the association of analog input voltages to the 2^n discrete digital result values (n bits resolution). Each digital result value (in the range of 0 to 2^n-1) represents an input voltage range defined by the reference voltage range divided by 2^n. This range (called quantization step or code width) represents the granularity (called LSB_n) of the ADC. The discrete character of the digital result generates a system-inherent quantization uncertainty of ±0.5 LSB_n for each conversion result.
The ideal transfer curve has the first digital transition (between 0 and 1) when the analog input reaches 0.5 LSB_n. The quantization steps are equally distributed over the input voltage range.
Analog input voltages below or above the reference voltage limits lead to a saturation of the digital result at 0 or 2^n-1.

The real transfer curve can exhibit certain deviations from the ideal transfer curve:

- The **offset error** is the deviation of the real transfer line from the ideal transfer line at the lowest code. This refers to best-fit lines through all possible codes, for both cases.
- The **gain error** is the deviation of the slope of the real transfer line from the slope of the ideal transfer line. This refers to best-fit lines through all possible codes, for both cases.
- The **differential non-linearity error** (DNL) is the deviation of the real code width (variation of the analog input voltage between two adjacent digital conversion results) from the ideal code width.
- The **integral non-linearity error** (INL) is the deviation of the real transfer curve from an adjusted ideal transfer curve (same offset and gain error as the real curve, but equal code widths).
- The **total unadjusted error** (TUE) describes the maximum deviation between a real conversion result and the ideal transfer characteristics over a given measurement range. Since some of these errors noted above can compensate each other, the TUE value generally is much less than the sum of the individual errors.
  The TUE also covers production process variations and internal noise effects (if switching noise is generated by the system, this generally leads to an increased TUE value).

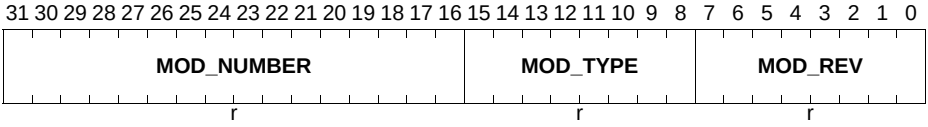
27.3 Configuration of General Functions

While many parameters can be selected individually for each channel, source, or group, some adjustments are valid for the whole ADC cluster.

27.3.1 Module Identification

The module identification register indicates the version of the ADC module that is used in the TC21x/TC22x/TC23x.

ID
Module Identification Register (0008$_H$) Reset Value: 00C5 C0XX$_H$



Field	Bits	Type	Description
MOD_REV	[7:0]	r	**Module Revision** Indicates the revision number of the implementation. This information depends on the design step.
MOD_TYPE	[15:8]	r	**Module Type** This internal marker is fixed to C0$_H$.
MOD_NUMBER	[31:16]	r	**Module Number** Indicates the module identification number (00C5$_H$ = SARADC).

27.3.2 System Registers

A set of standardized registers provides general access to the module and controls basic system functions.

The Clock Control Register **CLC** allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. Register **CLC** controls the module clock signal and the reactivity to the sleep mode signal.

CLC
Clock Control Register **(0000_H)** **Reset Value: 0000 0003_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**EDIS**	**0**	**DISS**	**DISR**
r	r	r	r	r	r	r	r	r	r	r	r	rw	r	r	rw

Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. Also the analog section is disabled by clearing ANONS. 0_B On request: enable the module clock 1_B Off request: stop the module clock
DISS	1	r	**Module Disable Status Bit** 0_B Module clock is enabled 1_B Off: module is not clocked
0	2	r	**Reserved, write 0, read as 0**
EDIS	3	rw	**Sleep Mode Enable Control** Used to control module's reaction to sleep mode. 0_B Sleep mode request is enabled and functional 1_B Module disregards the sleep mode control signal
0	[31:4]	r	**Reserved, write 0, read as 0**

The OCDS control and status register OCS controls the module's behavior in suspend mode (used for debugging) and includes the module-related control bits for the OCDS Trigger Bus (OTGB).

The OCDS Control and Status (OCS) register is cleared by Debug Reset.
The OCS register can only be written when the OCDS is enabled.
If OCDS is being disabled, the OCS register value will not change.
When OCDS is disabled the OCS suspend control is ineffective.
Write access is 32 bit wide only and requires Supervisor Mode.

OCS

OCDS Control and Status Register (0028_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**SUS STA**	**SUS _P**	**SUS**				**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	rh	w	rw				r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**TG _P**	**TGB**	**TGS**	
r	r	r	r	r	r	r	r	r	r	r	r	w	rw	rw	

Field	Bits	Type	Description
TGS	[1:0]	rw	**Trigger Set for OTGB0/1** 00_B No Trigger Set output 01_B Trigger Set 1: TS16_SSIG, input sample signals 10_B Reserved 11_B Reserved
TGB	2	rw	**OTGB0/1 Bus Select** 0_B Trigger Set is output on OTGB0 1_B Trigger Set is output on OTGB1
TG_P	3	w	**TGS, TGB Write Protection** TGS and TGB are only written when TG_P is 1, otherwise unchanged. Read as 0.
0	[23:4]	r	**Reserved, write 0, read as 0**

Field	Bits	Type	Description
SUS	[27:24]	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0000_B Will not suspend 0001_B Hard suspend: Clock is switched off immediately. 0010_B Soft suspend mode 0: Stop conversions after the currently running one is completed and its result has been stored. No change for the arbiter. 0011_B Soft suspend mode 1: Stop conversions after the currently running one is completed and its result has been stored. Stop arbiter after the current arbitration round. others: Reserved
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	[31:30]	r	**Reserved, write 0, read as 0**

Table 27-2 TS16_SSIG Trigger Set VADC

Bits	Name	Description
[1:0]	GxSAMPLE	Input signal sample phase of converter group x (x = 1-0)
[3:2]	GxSAMPLE	Sample phase of group x (x = 3-2), in the ADAS device
[15:4]	0	Reserved

The Access Enable Register 0 controls write access[1)] for transactions with the on-chip bus master TAG IDs 00 0000_B to 01 1111_B (see On-Chip Bus chapter for the mapping of product TAG ID <-> master peripheral). Register provides one enable bit for each possible TAG ID encoding.
Register ACCEN0 itself is protected by the Safety Endinit feature.

Mapping of TAG IDs to .ENx: EN0 -> TAG ID 00 0000_B, EN1 -> TAG ID 00 0001_B , ... , EN31 -> TAG ID 01 1111_B (TAG IDs 1X $XXXX_B$ are not used).

ACCEN0
Access Enable Register 0 ($003C_H$) Reset Value: FFFF FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN 31	**EN 30**	**EN 29**	**EN 28**	**EN 27**	**EN 26**	**EN 25**	**EN 24**	**EN 23**	**EN 22**	**EN 21**	**EN 20**	**EN 19**	**EN 18**	**EN 17**	**EN 16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN 15	**EN 14**	**EN 13**	**EN 12**	**EN 11**	**EN 10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENx (x = 0-31)	x	rw	**Access Enable for Master TAG ID x** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID x 0_B No Write access 1_B Write access will be executed

Individual Module Reset

The Kernel Reset Registers **KRST0/KRST1** are used to reset the related module kernel. To reset a module kernel it is necessary to set the RST bits in both Kernel Reset Registers. The RST bits will be cleared by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Registers 0 and 1 each include bit RST. To trigger a module reset, both RST bits must be set. They will be cleared automatically with the end of the BPI kernel reset sequence.

1) The Access Enable functionality controls only write transactions to registers CLC, OCS, KRSTx, and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers

Kernel Reset Register 0 includes a kernel reset status bit that is set in the same clock cycle the RST bit is cleared. This bit indicates that a kernel reset was processed. Bit RSTSTAT can be cleared by setting bit CLR in register **KRSTCLR**.

Note: During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

KRST0
Kernel Reset Register 0 **(0034_H)** **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**RSTSTAT**	**RST**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	rh	rwh

Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** Request a kernel reset. The reset is executed if the reset bits of both kernel reset registers are set. 0_B No action 1_B A kernel reset was requested RST is cleared after the kernel reset was executed.
RSTSTAT	1	rh	**Kernel Reset Status** Indicates an executed kernel reset. RSTSTAT is set after the execution of a kernel reset in the same clock cycle in which the reset bits are cleared. 0_B No kernel reset was executed 1_B Kernel reset was executed Clear RSTSTAT by setting bit CLR in register KRSTCLR.
0	[31:2]	r	**Reserved, write 0, read as 0**

KRST1
Kernel Reset Register 1 (0030_H) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**RST**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rwh

Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** Request a kernel reset. The reset is executed if the reset bits of both kernel reset registers are set. 0_B No action 1_B A kernel reset was requested RST is cleared after the kernel reset was executed.
0	[31:1]	r	**Reserved, write 0, read as 0**

The Kernel Reset Status Clear register is used to clear the Kernel Reset Status bit (RSTSTAT).

KRSTCLR
Kernel Reset Status Clear Register ($002C_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**CLR**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	w

Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT Read always as 0.
0	[31:1]	r	**Reserved, write 0, read as 0**

27.3.3 General Clocking Scheme and Control

The A/D Converters of the TC21x/TC22x/TC23x are supplied with a global clock signal from the system f_{ADC}. Two clock signals are derived from this input and are distributed to all converters. The global configuration register defines common clock bases for all converters of the cluster. This ensures deterministic behavior of converters that shall operate in parallel.

The analog converter clock f_{ADCI} determines the performance of the converters and must be selected to comply with the specification given in the Data Sheet.

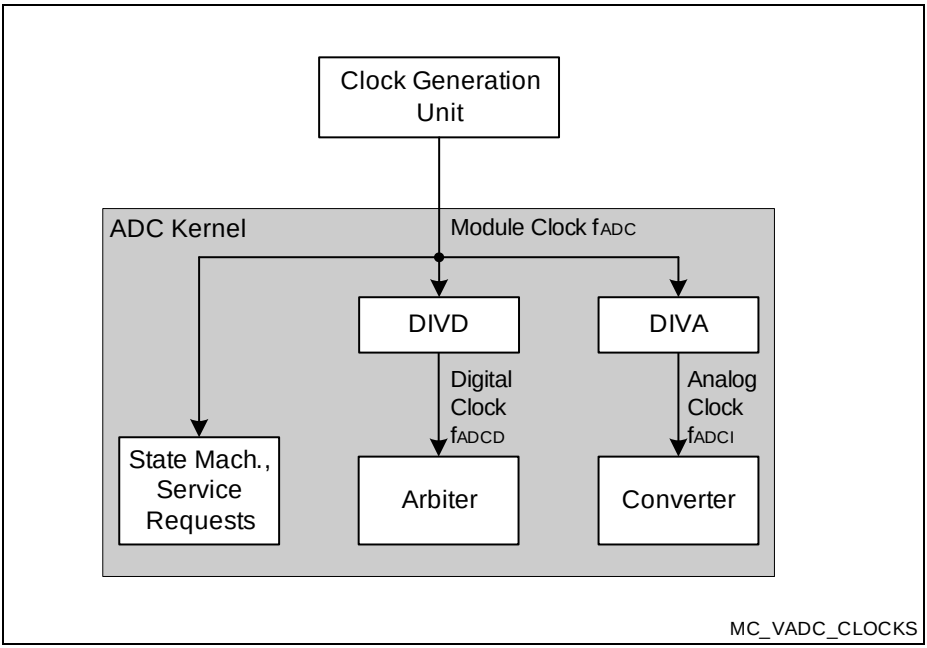


Figure 27-7 Clock Signal Summary

Global Configuration

The global configuration register GLOBCFG provides global control and configuration options that are valid for all converters of the cluster.

GLOBCFG
Global Configuration Register (0080_H) Reset Value: $0000\ 000F_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SUCAL	0	0	0	0	0	0	0	0	0	0	0	DPCAL3	DPCAL2	DPCAL1	DPCAL0
w	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIVWC	LOSUP	0	REFPC	0	0	DIVD		DCMSB	0	0	DIVA				
w	rw	r	rw	r	r	rw		rw	r	r	rw				

Field	Bits	Type	Description
DIVA	[4:0]	rw	**Divider Factor for the Analog Internal Clock** Defines the frequency of the basic converter clock f_{ADCI} (base clock for conversion and sample phase). 00_H $f_{ADCI} = f_{ADC} / 2$ 01_H $f_{ADCI} = f_{ADC} / 2$ 02_H $f_{ADCI} = f_{ADC} / 3$ ... $1F_H$ $f_{ADCI} = f_{ADC} / 32$
0	[6:5]	r	**Reserved, write 0, read as 0**
DCMSB	7	rw	**Double Clock for the MSB Conversion** Selects an additional clock cycle for the conversion step of the MSB.[1)] 0_B 1 clock cycles for the MSB (standard) 1_B 2 clock cycles for the MSB (f_{ADCI} > 20 MHz)
DIVD	[9:8]	rw	**Divider Factor for the Arbiter Clock** Defines the frequency of the arbiter clock f_{ADCD}. 00_B $f_{ADCD} = f_{ADC}$ 01_B $f_{ADCD} = f_{ADC} / 2$ 10_B $f_{ADCD} = f_{ADC} / 3$ 11_B $f_{ADCD} = f_{ADC} / 4$
0	[11:10]	r	**Reserved, write 0, read as 0**

Field	Bits	Type	Description
REFPC	12	rw	**Reference Precharge Control** 0_B Use V_{DDM} for precharging and V_{AREF} for the final adjustment during a conversion 1_B Use V_{AREF} for the conversion
0	13	r	**Reserved, write 0, read as 0**
LOSUP	14	rw	**Low Power Supply Voltage Select** Adjusts the analog circuitry to the supply voltage used in the application system. 0_B 5 V power supply is connected 1_B 3.3 V power supply is connected *Note: Make sure to keep LOSUP = 0 in the case of a 5 V supply.*
DIVWC	15	w	**Write Control for Divider Parameters** 0_B No write access to divider parameters 1_B Bitfields DIVA, DCMSB, DIVD, REFPC, LOSUP can be written
DPCALx (x = 0 - 3)	x+16	rw	**Disable Post-Calibration**[2] 0_B Automatic post-calibration after each conversion of group x 1_B No post-calibration
0	[30:18]	r	**Reserved, write 0, read as 0**
SUCAL	31	w	**Start-Up Calibration** The 0-1 transition of bit SUCAL initiates the start-up calibration phase of all calibrated analog converters. 0_B No action 1_B Initiate the start-up calibration phase (indication in bit GxARBCFG.CAL)

1) Please also refer to section **"Conversion Timing" on Page 27-109**.

2) The standard device only provides 2 converters, so only the lower 2 bits are valid.

Priority Channel Assignment

Each channel of a group can be assigned to this group's request sources and is then regarded as a priority channel. An assigned priority channel can only be converted by its own group's request sources. A not assigned channel can also be converted by the background request source.

GxCHASS (x = 0 - 3)
Channel Assignment Register, Group x
($x * 0400_H + 0488_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ASSCH31	**ASSCH30**	**ASSCH29**	**ASSCH28**	**ASSCH27**	**ASSCH26**	**ASSCH25**	**ASSCH24**	**ASSCH23**	**ASSCH22**	**ASSCH21**	**ASSCH20**	**ASSCH19**	**ASSCH18**	**ASSCH17**	**ASSCH16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ASSCH15	**ASSCH14**	**ASSCH13**	**ASSCH12**	**ASSCH11**	**ASSCH10**	**ASSCH9**	**ASSCH8**	**ASSCH7**	**ASSCH6**	**ASSCH5**	**ASSCH4**	**ASSCH3**	**ASSCH2**	**ASSCH1**	**ASSCH0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ASSCHy (y = 0 - 31)	y	rw	**Assignment for Channel y** 0_B Channel y can be a background channel converted with lowest priority 1_B Channel y is a priority channel within group x

Priority Result Register Assignment

Each result register of a group can be assigned to this group's request sources and is then regarded as a reserved resource. An assigned result register can only be written by its own group's request sources. A not assigned result register can also be written by the background request source.

GxRRASS (x = 0 - 3)
Result Assignment Register, Group x

(x * 0400_H + $048C_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ASSRR15	**ASSRR14**	**ASSRR13**	**ASSRR12**	**ASSRR11**	**ASSRR10**	**ASSRR9**	**ASSRR8**	**ASSRR7**	**ASSRR6**	**ASSRR5**	**ASSRR4**	**ASSRR3**	**ASSRR2**	**ASSRR1**	**ASSRR0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ASSRRy (y = 0 - 15)	y	rw	**Assignment for Result Register y** 0_B Result register y can also be written by the background source 1_B Result register y can only be written by sources within group x
0	[31:16]	r	**Reserved, write 0, read as 0**

27.3.4 Register Access Control

Several protection schemes are provided to prevent unintended write access to control bitfields of the VADC.

- The registers of the VADC are protected by the general access control mechanism that is configured by register **ACCEN0**.
- A specific register access control scheme provides a versatile protection scheme against unintended corruption of register contents. Registers ACCPROT0/1 allow the restriction of write accesses for several groups of registers. The registers to be protected can be selected by the user. **Table 27-3** lists the registers that belong to each register group.
  Registers ACCPROT0/1 themselves are protected by the Safety Endinit feature.
- Groups of bitfields within a register may also be protected by an associated write control bit. This write control bit (xxWC) must be written with 1 along with the write access to the intended bitfield(s).

ACCPROT0
Access Protection Register **(0088_H)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APGC	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**API3**	**API2**	**API1**	**API0**
rw	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
APEM	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**APC3**	**APC2**	**APC1**	**APC0**
rw	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw

Field	Bits	Type	Description
APCx (x = 0 - 3)	x	rw	**Access Protection Channel Control, Group 0 - 3[1)]** 0_B Full access to registers 1_B Write access to channel control registers is blocked
0	[14:2]	r	**Reserved, write 0, read as 0**
APEM	15	rw	**Access Protection External Multiplexer** 0_B Full access to registers 1_B Write access to external multiplexer registers is blocked

Field	Bits	Type	Description
APIx (x = 0 - 3)	16 + x	rw	**Access Protection Initialization, Group 0 - 3[1)]** 0_B Full access to registers 1_B Write access to initialization registers is blocked
0	[30:18]	r	**Reserved, write 0, read as 0**
APGC	31	rw	**Access Protection Global Configuration** 0_B Full access to register 1_B Write access to global configuration register is blocked

1) The standard device only provides 2 converters, so only the lower 2 bits are valid.

ACCPROT1
Access Protection Register **($008C_H$)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**APR3**	**APR2**	**APR1**	**APR0**
r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
APTF	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**APS3**	**APS2**	**APS1**	**APS0**
rw	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw

Field	Bits	Type	Description
APSx (x = 0 - 3)	x	rw	**Access Protection Service Request, Group 0 - 3[1)]** 0_B Full access to registers 1_B Write access to service request registers is blocked
0	[14:2]	r	**Reserved, write 0, read as 0**
APTF	15	rw	**Access Protection Test Function** 0_B Full access to register 1_B Write access to test function register is blocked
APRx (x = 0 - 3)	16 + x	rw	**Access Protection Result Registers, Group 0 - 3[1)]** 0_B Full access to registers 1_B Write access to result registers is blocked

Field	Bits	Type	Description
0	[31:18]	r	**Reserved, write 0, read as 0**

1) The standard device only provides 2 converters, so only the lower 2 bits are valid.

Table 27-3 Register Protection Groups

Control Bits	Registers	Notes
APCx	GxCHCTR0 … GxCHCTRy	Channel control
APEM	EMUXSEL, GxEMUXCTR	External multiplexer control
APIx	GxARBCFG, GxARBPR, GxCHASS, GxRRASS, GxICLASS0/1, GxSYNCTR	Initialization
APGC	GLOBCFG	Gobal configuration
APSx	GxSEFLAG, GxSEVNP, GxCEFLAG, GxCEVNP0/1, GxREFLAG, GxREVNP0/1, GxSRACT	Service request control
APTF	GLOBTF, GLOBTE	Test functions
APRx	GxRCR0 … GxRCR15, GxBOUND, GxRES0 … GxRES15	Result control

27.4 Analog Module Activation and Control

The analog converter of the ADC is the functional block that generates the digital result values from the selected input voltage. It draws a permanent current during its operation and can be deactivated between conversions to reduce the consumed overall energy.

Note: After reset the analog converters are off. They must be enabled before triggering any action involving a converter.

The accuracy of the conversions is improved by calibrating the analog converter to compensate process, temperature, and voltage variations.

27.4.1 Analog Converter Control

The operating mode is determined by bitfield **GxARBCFG (x = 0 - 3)**.ANONS:

- ANONS = 11_B: **Normal Operation**
  The converter is active, conversions are started immediately.
  Requires no wakeup time.
- ANONS = 10_B or 01_B: **Reserved**
- ANONS = 00_B: **Converter switched Off** (default after reset)
  The converter is switched off. Furthermore, digital logic blocks are set to their initial state. If the arbiter is currently running, it completes the actual arbitration round and then stops.
  Before starting a conversion, select the active mode for ANONS.
  Requires the wakeup time (see below).

Wakeup Time from Analog Powerdown

When the converter is activated, it needs a certain wakeup time to settle before a conversion can be properly executed. This wakeup time can be established by waiting the required period before starting a conversion, or by adding it to the intended sample time.

The wakeup time is approximately 15 µs.
Exact numbers can be found in the respective Data Sheets.

Note: The wakeup time is also required after initially enabling the converter.

27.4.2 Calibration

Calibration automatically compensates deviations caused by process, temperature, and voltage variations. This ensures precise results throughout the operation time.

An initial start-up calibration is required once after a reset for all calibrated converters and is triggered globally. All calibrated converters must be enabled (ANONS = 11_B) before initiating the start-up calibration. Conversions may be started after the initial calibration sequence. This is indicated by bit CAL = 0_B.

After that, postcalibration cycles will compensate the effects of drifting parameters.

27.5 Conversion Request Generation

The conversion request unit of a group autonomously handles the generation of conversion requests. Three request sources (2 group-specific sources and the background source) can generate requests for the conversion of an analog channel. The arbiter resolves concurrent requests and selects the channel to be converted next.

Upon a trigger event, the request source requests the conversion of a certain analog input channel or a sequence of channels.

- **Software triggers**
  directly activate the respective request source.
- **External triggers**
  synchronize the request source activation with external events, such as a trigger pulse from a timer generating a PWM signal or from a port pin.

Application software selects the trigger type and source, the channel(s) to be converted, and the request source priority. A request source can also be activated directly by software without requiring an external trigger.

The arbiter regularly scans the request sources for pending conversion requests and selects the conversion request with the highest priority. This conversion request is then forwarded to the converter to start the sampling and conversion of the requested channel.

Each request source can operate in single-shot or in continuous mode:

- **In single-shot mode**,
  the programmed conversion (sequence) is requested once after being triggered. A subsequent conversion (sequence) must be triggered again.
- **In continuous mode**,
  the programmed conversion (sequence) is automatically requested repeatedly after being triggered once.

For each request source of a group, external triggers are generated from one of 15 selectable trigger inputs (REQTRx[O:A]) and from one of 16 selectable gating inputs (REQGTx[P:A])[1)]. The available trigger signals for the TC21x/TC22x/TC23x are listed in **Section 27.14.4**.

*Note: **Figure 27-3 "Conversion Request Unit" on Page 27-6** summarizes the request sources.*

1) The selected gating input can be used as a trigger signal (REQTRxP).

Two types of requests sources are available:

- **A queued source** can issue conversion requests for an arbitrary sequence of input channels. The channel numbers for this sequence can be freely programmed[1]. This supports application-specific conversion sequences that cannot be covered by a channel scan source. Also, multiple conversions of the same channel within a sequence are supported.
  A queued source converts a series of input channels permanently or on a regular time base. For example, if programmed with medium priority, some input channels can be converted upon a specified event (e.g. synchronized to a PWM). Conversions of lower priority sources are suspended in the meantime.
  – Request source 0 is a group-specific 8-stage queued source.
  – Request source 3 is a group-specific 8-stage queued source with test features.
- **A channel scan source** can issue conversion requests for a coherent sequence of input channels. This sequence begins with the highest enabled channel number and continues towards lower channel numbers. All available channels[1] can be enabled for the scan sequence. Each channel is converted once per sequence.
  A scan source converts a series of input channels permanently or on a regular time base. For example, if programmed with low priority, some input channels can be scanned in a background task to update information that is not time-critical.
  – Request source 1 is a group-specific channel scan source.
  – Request source 2 is a global channel scan source (background source).
  The background source can request conversions of all channels of all groups.

1) The availablity of input channels depends on the package of the used product type. A summary can be found in **Section 27.14.3**.
The background source can only request non-priority channels, i.e. channels that are not selected in registers GxCHASS. Priority channels are reserved for the group-specific request sources 0 and 1.

27.5.1 Queued Request Source Handling

A queued request source supports short conversion sequences (up to 8) of arbitrary channels (contrary to a scan request source with a fixed conversion order for the enabled channels). The programmed sequence is stored in a queue buffer (based on a FIFO mechanism). The requested channel numbers are entered via the queue input, while queue stage 0 defines the channel to be converted next.

A conversion request is only issued to the request source arbiter if a valid entry is stored in queue stage 0.

If the arbiter aborts a conversion triggered by a queued request source due to higher priority requests, the corresponding conversion parameters are automatically saved in the backup stage. This ensures that an aborted conversion is not lost but takes part in the next arbitration round (before stage 0).

The trigger and gating unit generates trigger events from the selected external (outside the ADC) trigger and gating signals. For example, a timer unit can issue a request signal to synchronize conversions to PWM events.

Trigger events start a queued sequence and can be generated either via software or via the selected hardware triggers. The occurrence of a trigger event is indicated by bit QSRx.EV. This flag is cleared when the corresponding conversion is started or by writing to bit QMRx.CEV.

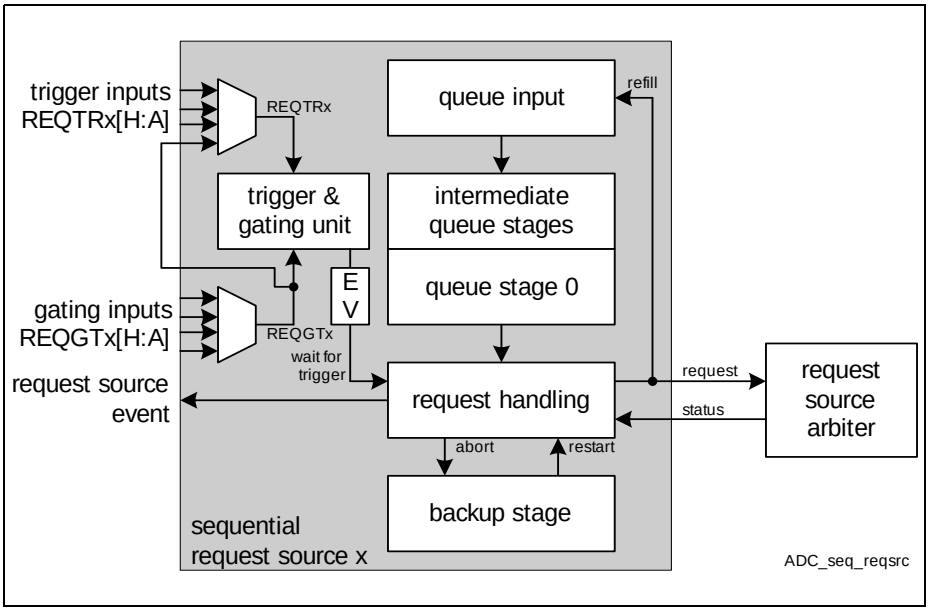


Figure 27-8 Queued Request Source

A sequence is defined by entering conversion requests into the queue input register (**GxQINR0 (x = 0 - 3)**). Each entry selects the channel to be converted and can enable an external trigger, generation of an interrupt, and an automatic refill (i.e. copy this entry to the top of the queue after conversion). The entries are stored in the queue buffer stages.

The content of stage 0 (**GxQ0R0 (x = 0 - 3)**) selects the channel to be converted next. When the requested conversion is started, the contents of this queue stage is invalidated and copied to the backup stage. Then the next queue entry can be handled (if available).

Note: The contents of the queue stages cannot be modified directly, but only by writing to the queue input or by flushing the queue.
*The current status of the queue is shown in register **GxQSR0 (x = 0 - 3)**.*
If all queue entries have automatic refill selected, the defined conversion sequence can be repeated without re-programming.

Properties of the Queued Request Source

Queued request source 0 provides 8 buffer stages and can handle sequences of up to 8 input channel entries. It supports short application-specific conversion sequences, especially for timing-critical sequences containing also multiple conversions of the same channel.

Queued Source Operation

Configure the queued request source by executing the following actions:

- Define the sequence by writing the entries to the queue input **GxQINR0 (x = 0 - 3)**. Initialize the complete sequence before enabling the request source, because with enabled refill feature, software writes to QINRx are not allowed.
- If hardware trigger or gating is desired, select the appropriate trigger and gating inputs and the proper transitions by programming **GxQCTRL0 (x = 0 - 3)**.
  Enable the trigger and select the gating mode by programming bitfield ENGT in register **GxQMR0 (x = 0 - 3)**.[1)]
- Enable the corresponding arbitration slot (0) to accept conversion requests from the queued source (see register **GxARBPR (x = 0 - 3)**).

Start a queued sequence by generating a trigger event:

- If a hardware trigger is selected and enabled, generate the configured transition at the selected input signal, e.g. from a timer or an input pin.
- Generate a software trigger event by setting GxQMR0.TREV = 1.

1) To avoid unexpected signal transitions, select/enable the trigger and/or gate inputs before activating the corresponding request source. If PDOUT signals from the ERU are used, first initialize the ERU.

- Write a new entry to the queue input of an empty queue. This leads to a (new) valid queue entry that is forwarded to queue stage 0 and starts a conversion request (if enabled by GxQMR0.ENGT and without waiting for an external trigger).

Note: If the refill mechanism is activated, a processed entry is automatically reloaded into the queue. This permanently repeats the respective sequence (autoscan). In this case, do not write to the queue input while the queued source is running. Write operations to a completely filled queue are ignored.

Stop or abort an ongoing queued sequence by executing the following actions:

- If external gating is enabled, switch the gating signal to the defined inactive level. This does not modify the queue entries, but only prevents issuing conversion requests to the arbiter.
- Disable the corresponding arbitration slot (0) in the arbiter. This does not modify the queue entries, but only prevents the arbiter from accepting requests from the request handling block.
- Disable the queued source by clearing bitfield ENGT = 00_B.
  - Invalidate the next pending queue entry by setting bit GxQMR0.CLRV = 1. If the backup stage contains a valid entry, this one is invalidated, otherwise stage 0 is invalidated.
  - Remove all entries from the queue by setting bit GxQMR0.FLUSH = 1.

Queue Request Source Events and Service Requests

A request source event of a queued source occurs when a conversion is finished. A source event service request can be generated based on a request source event according to the structure shown in **Figure 27-9**. If a request source event is detected, it sets the corresponding indication flag in register **GxSEFLAG (x = 0 - 3)**. These flags can also be set by writing a 1 to the corresponding bit position, whereas writing 0 has no effect. The indication flags can be cleared by SW by writing a 1 to the corresponding bit position in register **GxSEFCLR (x = 0 - 3)**.
The interrupt enable bit is taken from stage 0 for a normal sequential conversion, or from the backup stage for a repeated conversion after an abort.

The service request output line SRx that is selected by the request source event interrupt node pointer bitfields in register **GxSEVNP (x = 0 - 3)** becomes activated each time the related request source event is detected (and enabled by GxQ0R0.ENSI, or GxQBUR0.ENSI respectively) or the related bit position in register **GxSEFLAG (x = 0 - 3)** is written with a 1 (this write action simulates a request source event).

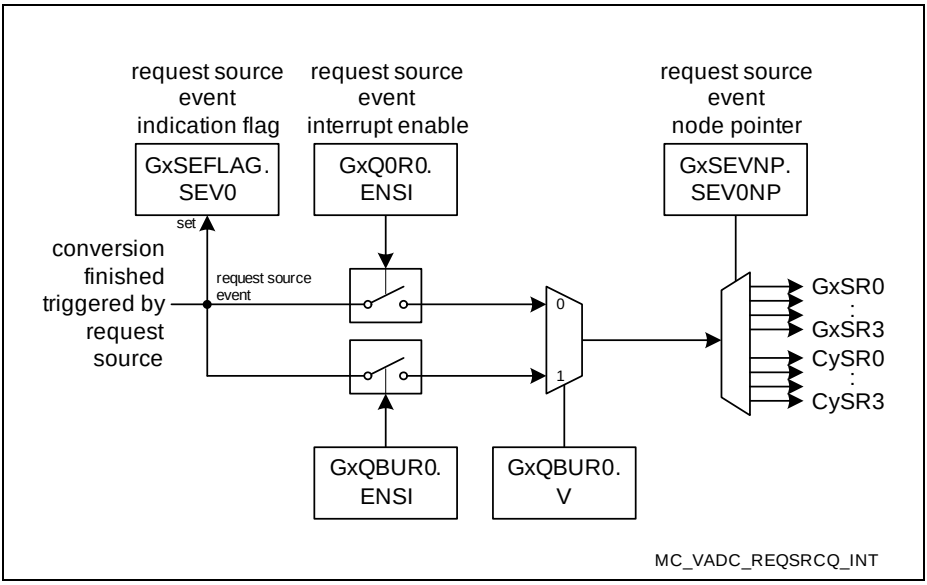


Figure 27-9 Interrupt Generation of a Queued Request Source

The control register of the queue source selects the external gate and/or trigger signals. Write control bits allow separate control of each function with a simple write access.

GxQCTRL0 (x = 0 - 3)
Queue 0 Source Control Register, Group x
($x * 0400_H + 0500_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TM WC	0	0	TM EN	0	0	0	0	GT WC	0	0	GT LVL	GT SEL			
w	r	r	rw	r	r	r	r	w	r	r	rh	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XT WC	XT MODE		XT LVL	XT SEL				0	0	0	0	SRCRESREG			
w	rw		rh	rw				r	r	r	r	rw			

Field	Bits	Type	Description
SRCRESREG	[3:0]	rw	**Source-specific Result Register** 0000_B Use GxCHCTRy.RESREG to select a group result register 0001_B Store result in group result register GxRES1 … 1111_B Store result in group result register GxRES15
0	[7:4]	r	**Reserved, write 0, read as 0**
XTSEL	[11:8]	rw	**External Trigger Input Selection** The connected trigger input signals are listed in **Table 27-13 “Digital Connections in the TC21x/TC22x/TC23x” on Page 27-176** *Note: XTSEL = 1111_B uses the selected gate input as trigger source (ENGT must be $0X_B$).*
XTLVL	12	rh	**External Trigger Level** Current level of the selected trigger input
XTMODE	[14:13]	rw	**Trigger Operating Mode** 00_B No external trigger 01_B Trigger event upon a falling edge 10_B Trigger event upon a rising edge 11_B Trigger event upon any edge

Field	Bits	Type	Description
XTWC	15	w	**Write Control for Trigger Configuration** 0_B No write access to trigger configuration 1_B Bitfields XTMODE and XTSEL can be written
GTSEL	[19:16]	rw	**Gate Input Selection** The connected gate input signals are listed in **Table 27-13 “Digital Connections in the TC21x/TC22x/TC23x” on Page 27-176**
GTLVL	20	rh	**Gate Input Level** Current level of the selected gate input
0	[22:21]	r	**Reserved, write 0, read as 0**
GTWC	23	w	**Write Control for Gate Configuration** 0_B No write access to gate configuration 1_B Bitfield GTSEL can be written
0	[27:24]	r	**Reserved, write 0, read as 0**
TMEN	28	rw	**Timer Mode Enable** 0_B No timer mode: standard gating mechanism can be used 1_B Timer mode for equidistant sampling enabled: standard gating mechanism must be disabled
0	[30:29]	r	**Reserved, write 0, read as 0**
TMWC	31	w	**Write Control for Timer Mode** 0_B No write access to timer mode 1_B Bitfield TMEN can be written

The Queue Mode Register configures the operating mode of a queued request source.

GxQMR0 (x = 0 - 3)
Queue 0 Mode Register, Group x

($x * 0400_H + 0504_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**RPT DIS**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**CEV**	**FLU SH**	**TR EV**	**CLR V**	**0**	**0**	**0**	**0**	**0**	**EN TR**	**ENGT**	
r	r	r	r	w	w	w	w	r	r	r	r	r	rw	rw	

Field	Bits	Type	Description
ENGT	[1:0]	rw	**Enable Gate** Selects the gating functionality for source 0/2. 00_B No conversion requests are issued 01_B Conversion requests are issued if a valid conversion request is pending in the queue 0 register or in the backup register 10_B Conversion requests are issued if a valid conversion request is pending in the queue 0 register or in the backup register and REQGTx = 1 11_B Conversion requests are issued if a valid conversion request is pending in the queue 0 register or in the backup register and REQGTx = 0 *Note: REQGTx is the selected gating signal.*
ENTR	2	rw	**Enable External Trigger** 0_B External trigger disabled 1_B The selected edge at the selected trigger input signal REQTR generates the trigger event
0	[7:3]	r	**Reserved, write 0, read as 0**

Field	Bits	Type	Description
CLRV	8	w	**Clear Valid Bit** 0_B No action 1_B The next pending valid queue entry in the sequence and the event flag EV are cleared. If there is a valid entry in the queue backup register (QBUR.V = 1), this entry is cleared, otherwise the entry in queue register 0 is cleared.
TREV	9	w	**Trigger Event** 0_B No action 1_B Generate a trigger event by software
FLUSH	10	w	**Flush Queue** 0_B No action 1_B Clear all queue entries (including backup stage) and the event flag EV. The queue contains no more valid entry.
CEV	11	w	**Clear Event Flag** 0_B No action 1_B Clear bit EV
0	[15:12]	r	**Reserved, write 0, read as 0**
RPTDIS	16	rw	**Repeat Disable** 0_B A cancelled conversion is repeated 1_B A cancelled conversion is discarded
0	[31:17]	r	**Reserved, write 0, read as 0**

The Queue Status Register indicates the current status of the queued source. The filling level and the empty information refer to the queue intermediate stages (if available) and to the queue register 0. An aborted conversion stored in the backup stage is not indicated by these bits (therefore, see QBUR0.V).

GxQSR0 (x = 0 - 3)
Queue 0 Status Register, Group x

($x * 0400_H + 0508_H$) **Reset Value: 0000 0020$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**EV**	**REQ GT**	**0**	**EMP TY**	**0**	**FILL**			
r	r	r	r	r	r	r	rh	rh	r	rh	r	rh			

Field	Bits	Type	Description
FILL	[3:0]	rh	**Filling Level for Queue 0** Indicates the number of valid queue entries. It is incremented each time a new entry is written to QINR0 or by an enabled refill mechanism. It is decremented each time a requested conversion has been started. A new entry is ignored if the filling level has reached its maximum value. 0000_B There is 1 (if EMPTY = 0) or no (if EMPTY = 1) valid entry in the queue 0001_B There are 2 valid entries in the queue 0010_B There are 3 valid entries in the queue … 0111_B There are 8 valid entries in the queue others: Reserved
0	4	r	**Reserved, write 0, read as 0**
EMPTY	5	rh	**Queue Empty** 0_B There are valid entries in the queue (see FILL) 1_B No valid entries (queue is empty)
0	6	r	**Reserved, write 0, read as 0**

Field	Bits	Type	Description
REQGT	7	rh	**Request Gate Level** Monitors the level at the selected REQGT input. 0_B The gate input is low 1_B The gate input is high
EV	8	rh	**Event Detected** Indicates that an event has been detected while at least one valid entry has been in the queue (queue register 0 or backup stage). Once set, this bit is cleared automatically when the requested conversion is started. 0_B No trigger event 1_B A trigger event has been detected
0	[31:9]	r	**Reserved, write 0, read as 0**

The Queue Input Register is the entry point for conversion requests of a queued request source.

GxQINR0 (x = 0 - 3)
Queue 0 Input Register, Group x

($x * 0400_H + 0510_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**EXTR**	**ENSI**	**RF**	**REQCHNR**				
r	r	r	r	r	r	r	r	w	w	w			w		

Field	Bits	Type	Description
REQCHNR	[4:0]	w	**Request Channel Number** Defines the channel number to be converted
RF	5	w	**Refill** 0_B No refill: this queue entry is converted once and then invalidated 1_B Automatic refill: this queue entry is automatically reloaded into QINRx when the related conversion is started
ENSI	6	w	**Enable Source Interrupt** 0_B No request source interrupt 1_B A request source event interrupt is generated upon a request source event (related conversion is finished)
EXTR	7	w	**External Trigger** Enables the external trigger functionality. 0_B A valid queue entry immediately leads to a conversion request. 1_B A valid queue entry waits for a trigger event to occur before issuing a conversion request.
0	[31:8]	r	**Reserved, write 0, read as 0**

Note: Register QINR0 shares address with register QBUR0.
Write operations target the control bits in register QINR0. Read operations return the status bits from register QBUR0.

The queue registers 0 monitor the status of the pending request (queue stage 0).

GxQ0R0 (x = 0 - 3)
Queue 0 Register 0, Group x (x * 0400_H + $050C_H$) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**V**	**EXTR**	**ENSI**	**RF**	**REQCHNR**				
r	r	r	r	r	r	r	rh	rh	rh	rh	rh				

Field	Bits	Type	Description
REQCHNR	[4:0]	rh	**Request Channel Number** Stores the channel number to be converted.
RF	5	rh	**Refill** Selects the handling of handled requests. 0_B The request is discarded after the conversion start. 1_B The request is automatically refilled into the queue after the conversion start.
ENSI	6	rh	**Enable Source Interrupt** 0_B No request source interrupt 1_B A request source event interrupt is generated upon a request source event (related conversion is finished)
EXTR	7	rh	**External Trigger** Enables external trigger events. 0_B A valid queue entry immediately leads to a conversion request 1_B The request handler waits for a trigger event
V	8	rh	**Request Channel Number Valid** Indicates a valid queue entry in queue register 0. 0_B No valid queue entry 1_B The queue entry is valid and leads to a conversion request

Field	Bits	Type	Description
0	[31:9]	r	**Reserved, write 0, read as 0**

The Queue Backup Registers monitor the status of an aborted queued request.

GxQBUR0 (x = 0 - 3)
Queue 0 Backup Register, Group x

($x * 0400_H + 0510_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**V**	**EXTR**	**ENSI**	**RF**	**REQCHNR**				
r	r	r	r	r	r	r	rh	rh	rh	rh	rh				

Field	Bits	Type	Description
REQCHNR	[4:0]	rh	**Request Channel Number** The channel number of the aborted conversion that has been requested by this request source
RF	5	rh	**Refill** The refill control bit of the aborted conversion
ENSI	6	rh	**Enable Source Interrupt** The enable source interrupt control bit of the aborted conversion
EXTR	7	rh	**External Trigger** The external trigger control bit of the aborted conversion
V	8	rh	**Request Channel Number Valid** Indicates if the entry (REQCHNR, RF, TR, ENSI) in the queue backup register is valid. Bit V is set when a running conversion (that has been requested by this request source) is aborted, it is cleared when the aborted conversion is restarted. 0_B Backup register not valid 1_B Backup register contains a valid entry. This will be requested before a valid entry in queue register 0 (stage 0) will be requested.
0	[31:9]	r	**Reserved, write 0, read as 0**

Note: Register QBUR0 shares address with register QINR0.
Read operations return the status bits from register QBUR0. Write operations target the control bits in register QINR0.

Queued Request Source 3

Request source 3 operates in the same way as request source 0 except for the additional control bits for the automatic test functions. These test functions can be used if they are enabled via register **GLOBTE**.

Register **GxTRCTR (x = 0 - 3)** selects the additional internal trigger source.

Note: Queued request source 3 is available in groups 0 and 1.

The control register of the queue source selects the external gate and/or trigger signals. Write control bits allow separate control of each function with a simple write access.

Note: No difference to GxQCTRL0.

GxQCTRL3 (x = 0 - 3)
Queue 3 Source Control Register, Group x

($x * 0400_H + 0540_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TM WC	**0**	**0**	**TM EN**	**0**	**0**	**0**	**0**	**GT WC**	**0**	**0**	**GT LVL**	**GT SEL**			
w	r	r	rw	r	r	r	r	w	r	r	rh	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XT WC	**XT MODE**		**XT LVL**	**XT SEL**				**0**	**0**	**0**	**0**	**SRCRESREG**			
w	rw		rh	rw				r	r	r	r	rw			

Field	Bits	Type	Description
SRCRESREG	[3:0]	rw	**Source-specific Result Register** 0000_B Use GxCHCTRy.RESREG to select a group result register 0001_B Store result in group result register GxRES1 … 1111_B Store result in group result register GxRES15
0	[7:4]	r	**Reserved, write 0, read as 0**
XTSEL	[11:8]	rw	**External Trigger Input Selection** The connected trigger input signals are listed in **Table 27-13 “Digital Connections in the TC21x/TC22x/TC23x” on Page 27-176** *Note: XTSEL = 1111_B uses the selected gate input as trigger source (ENGT must be $0X_B$).*
XTLVL	12	rh	**External Trigger Level** Current level of the selected trigger input
XTMODE	[14:13]	rw	**Trigger Operating Mode** 00_B No external trigger 01_B Trigger event upon a falling edge 10_B Trigger event upon a rising edge 11_B Trigger event upon any edge

Field	Bits	Type	Description
XTWC	15	w	**Write Control for Trigger Configuration** 0_B No write access to trigger configuration 1_B Bitfields XTMODE and XTSEL can be written
GTSEL	[19:16]	rw	**Gate Input Selection** The connected gate input signals are listed in **Table 27-13 “Digital Connections in the TC21x/TC22x/TC23x” on Page 27-176** *Note: GTSEL = 1111_B uses the selected internal trigger source.*
GTLVL	20	rh	**Gate Input Level** Current level of the selected gate input
0	[22:21]	r	**Reserved, write 0, read as 0**
GTWC	23	w	**Write Control for Gate Configuration** 0_B No write access to gate configuration 1_B Bitfield GTSEL can be written
0	[27:24]	r	**Reserved, write 0, read as 0**
TMEN	28	rw	**Timer Mode Enable** 0_B No timer mode: standard gating mechanism can be used 1_B Timer mode for equidistant sampling enabled: standard gating mechanism must be disabled
0	[30:29]	r	**Reserved, write 0, read as 0**
TMWC	31	w	**Write Control for Timer Mode** 0_B No write access to timer mode 1_B Bitfield TMEN can be written

The Queue Mode Register configures the operating mode of a queued request source.

Note: No difference to GxQMR0.

GxQMR3 (x = 0 - 3)
Queue 3 Mode Register, Group x

($x * 0400_H + 0544_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**RPT DIS**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**CEV**	**FLU SH**	**TR EV**	**CLR V**	**0**	**0**	**0**	**0**	**0**	**EN TR**	**ENGT**	
r	r	r	r	w	w	w	w	r	r	r	r	r	rw	rw	

Field	Bits	Type	Description
ENGT	[1:0]	rw	**Enable Gate** Selects the gating functionality for source 0. 00_B No conversion requests are issued 01_B Conversion requests are issued if a valid conversion request is pending in the queue 0 register or in the backup register 10_B Conversion requests are issued if a valid conversion request is pending in the queue 0 register or in the backup register and REQGTx = 1 11_B Conversion requests are issued if a valid conversion request is pending in the queue 0 register or in the backup register and REQGTx = 0 *Note: REQGTx is the selected gating signal.*
ENTR	2	rw	**Enable External Trigger** 0_B External trigger disabled 1_B The selected edge at the selected trigger input signal REQTR generates the trigger event
0	[7:3]	r	**Reserved, write 0, read as 0**

Field	Bits	Type	Description
CLRV	8	w	**Clear Valid Bit** 0_B No action 1_B The next pending valid queue entry in the sequence and the event flag EV are cleared. If there is a valid entry in the queue backup register (QBUR.V = 1), this entry is cleared, otherwise the entry in queue register 0 is cleared.
TREV	9	w	**Trigger Event** 0_B No action 1_B Generate a trigger event by software
FLUSH	10	w	**Flush Queue** 0_B No action 1_B Clear all queue entries (including backup stage) and the event flag EV. The queue contains no more valid entry.
CEV	11	w	**Clear Event Flag** 0_B No action 1_B Clear bit EV
0	[15:12]	r	**Reserved, write 0, read as 0**
RPTDIS	16	rw	**Repeat Disable** 0_B A cancelled conversion is repeated 1_B A cancelled conversion is discarded
0	[31:17]	r	**Reserved, write 0, read as 0**

The Queue Status Register indicates the current status of the queued source. The filling level and the empty information refer to the queue intermediate stages (if available) and to the queue register 0. An aborted conversion stored in the backup stage is not indicated by these bits (therefore, see QBUR3.V).

Note: No difference to GxQSR0.

GxQSR3 (x = 0 - 3)
Queue 3 Status Register, Group x

(x * 0400_H + 0548_H) **Reset Value: $0000\ 0020_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**EV**	**REQ GT**	**0**	**EMP TY**	**0**	**FILL**			
r	r	r	r	r	r	r	rh	rh	r	rh	r	rh			

Field	Bits	Type	Description
FILL	[3:0]	rh	**Filling Level for Queue 3** Indicates the number of valid queue entries. It is incremented each time a new entry is written to QINR3 or by an enabled refill mechanism. It is decremented each time a requested conversion has been started. A new entry is ignored if the filling level has reached its maximum value. 0000_B There is 1 (if EMPTY = 0) or no (if EMPTY = 1) valid entry in the queue 0001_B There are 2 valid entries in the queue 0010_B There are 3 valid entries in the queue … 0111_B There are 8 valid entries in the queue others: Reserved
0	4	r	**Reserved, write 0, read as 0**
EMPTY	5	rh	**Queue Empty** 0_B There are valid entries in the queue (see FILL) 1_B No valid entries (queue is empty)
0	6	r	**Reserved, write 0, read as 0**

Field	Bits	Type	Description
REQGT	7	rh	**Request Gate Level** Monitors the level at the selected REQGT input. 0_B The gate input is low 1_B The gate input is high
EV	8	rh	**Event Detected** Indicates that an event has been detected while at least one valid entry has been in the queue (queue register 0 or backup stage). Once set, this bit is cleared automatically when the requested conversion is started. 0_B No trigger event 1_B A trigger event has been detected
0	[31:9]	r	**Reserved, write 0, read as 0**

The Queue Input Register is the entry point for conversion requests of a queued request source. QINR3 contains additional control bits to select the test functions. Please refer to **Section 27.11.4**.

*Note: The test functions selected via GxQINR3 only become active if enabled in register **GLOBTE**.*

GxQINR3 (x = 0 - 3)
Queue 3 Input Register, Group x

(x * 0400_H + 0550_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CD SEL		CD EN	MD PU	MD PD	PDD	0	EX TR	EN SI	RF	REQCHNR				
r	w		w	w	w	w	r	w	w	w	w				

Field	Bits	Type	Description
REQCHNR	[4:0]	w	**Request Channel Number** Defines the channel number to be converted
RF	5	w	**Refill** 0_B No refill: this queue entry is converted once and then invalidated 1_B Automatic refill: this queue entry is automatically reloaded into TQINR when the related conversion is started
ENSI	6	w	**Enable Source Interrupt** 0_B No request source interrupt 1_B A request source event interrupt is generated upon a request source event (related conversion is finished)
EXTR	7	w	**External Trigger** Enables the external trigger functionality. 0_B A valid queue entry immediately leads to a conversion request. 1_B A valid queue entry waits for a trigger event to occur before issuing a conversion request.

Field	Bits	Type	Description
0	8	r	**Reserved, write 0, read as 0**
PDD	9	w	**Pull-Down Diagnostics Enable** 0_B Disconnected 1_B The pull-down diagnostics device is active *Note: Channels with pull-down diagnostics device are marked in **Table 27-12**.*
MDPD, MDPU	10, 11	w	**Multiplexer Diagnostics Pull-Devices Enable** 0_B Disconnected 1_B The respective device is active Connecting combinations of pull-up and/or pull-down devices generate various loads for testing. *Note: Channels with multiplexer diagnostics pull devices are marked in **Table 27-12**.*
CDEN	12	w	**Converter Diagnostics Enable** 0_B All diagnostic pull devices are disconnected 1_B Diagnostic pull devices connected as selected by bitfield CDSEL
CDSEL	[14:13]	w	**Converter Diagnostics Pull-Devices Select** 00_B Connected to VAREF 01_B Connected to VAGND 10_B Connected to 1/3rd VAREF 11_B Connected to 2/3rd VAREF
0	[31:15]	r	**Reserved, write 0, read as 0**

Note: Register QINR3 shares address with register QBUR3.
Write operations target the control bits in register QINR3. Read operations return the status bits from register QBUR3.

The queue register 0 monitors the status of the pending request (queue stage 0).

GxQ0R3 (x = 0 - 3)
Queue 3 Register 0, Group x ($x * 0400_H + 054C_H$) Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**CD SEL**		**CD EN**	**MD PU**	**MD PD**	**PDD**	**V**	**EX TR**	**EN SI**	**RF**	**REQCHNR**				
r	rh		rh	rh	rh	rh	rh	rh	rh	rh	rh				

Field	Bits	Type	Description
REQCHNR	[4:0]	rh	**Request Channel Number** Indicates the channel number to be converted.
RF	5	rh	**Refill** Indicates the handling of handled requests. 0_B The request is discarded after the conversion start. 1_B The request is automatically refilled into the queue after the conversion start.
ENSI	6	rh	**Enable Source Interrupt** 0_B No request source interrupt 1_B A request source event interrupt is generated upon a request source event (related conversion is finished)
EXTR	7	rh	**External Trigger** Enables external trigger events. 0_B A valid queue entry immediately leads to a conversion request 1_B The request handler waits for a trigger event
V	8	rh	**Request Channel Number Valid** Indicates a valid queue entry in queue register 0. 0_B No valid queue entry 1_B The queue entry is valid and leads to a conversion request

Field	Bits	Type	Description
PDD	9	rh	**Pull-Down Diagnostics Enable** Indicates the activation of the strong pull-down. 0_B Disconnected 1_B The pull-down diagnostics device is active
MDPD, MDPU	10, 11	rh	**Multiplexer Diagnostics Pull-Devices Enable** Indicates the activation of multiplexer diagnostics pull-devices. 0_B Disconnected 1_B The respective device is active
CDEN	12	rh	**Converter Diagnostics Enable** Indicates the activation of converter diagnostics pull-devices. 0_B All diagnostic pull devices are disconnected 1_B Diagnostic pull devices connected as selected by bitfield CDSEL
CDSEL	[14:13]	rh	**Converter Diagnostics Pull-Devices Select** Indicates the selected multiplexer diagnostics pull-devices. 00_B Connected to VAREF 01_B Connected to VAGND 10_B Connected to 1/3rd VAREF 11_B Connected to 2/3rd VAREF
0	[31:15]	r	**Reserved, write 0, read as 0**

The Queue Backup Register monitors the status of an aborted queued request.

GxQBUR3 (x = 0 - 3)
Queue 3 Backup Register, Group x

($x * 0400_H + 0550_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CDSEL		CDEN	MDPU	MDPD	PDD	V	EXTR	ENSI	RF	REQCHNR				
r	rh		rh	rh	rh	rh	rh	rh	rh	rh	rh				

Field	Bits	Type	Description
REQCHNR	[4:0]	rh	**Request Channel Number** The channel number of the aborted conversion that has been requested by this request source
RF	5	rh	**Refill** The refill control bit of the aborted conversion
ENSI	6	rh	**Enable Source Interrupt** The enable source interrupt control bit of the aborted conversion
EXTR	7	rh	**External Trigger** The external trigger control bit of the aborted conversion
V	8	rh	**Request Channel Number Valid** Indicates if the entry (REQCHNR, RF, TR, ENSI) in the queue backup register is valid. Bit V is set when a running conversion (that has been requested by this request source) is aborted, it is cleared when the aborted conversion is restarted. 0_B Backup register not valid 1_B Backup register contains a valid entry. This will be requested before a valid entry in queue register 0 (stage 0) will be requested.

Field	Bits	Type	Description
PDD	9	rh	**Pull-Down Diagnostics Enable** Indicates the activation of the strong pull-down. 0_B Disconnected 1_B The pull-down diagnostics device is active
MDPD, MDPU	10, 11	rh	**Multiplexer Diagnostics Pull-Devices Enable** Indicates the activation of multiplexer diagnostics pull-devices. 0_B Disconnected 1_B The respective device is active
CDEN	12	rh	**Converter Diagnostics Enable** Indicates the activation of converter diagnostics pull-devices. 0_B All diagnostic pull devices are disconnected 1_B Diagnostic pull devices connected as selected by bitfield CDSEL
CDSEL	[14:13]	rh	**Converter Diagnostics Pull-Devices Select** Indicates the selected multiplexer diagnostics pull-devices. 00_B Connected to VAREF 01_B Connected to VAGND 10_B Connected to 1/3rd VAREF 11_B Connected to 2/3rd VAREF
0	[31:15]	r	**Reserved, write 0, read as 0**

Note: Register QBUR3 shares address with register QINR3.
Read operations return the status bits from register QBUR3. Write operations target the control bits in register QINR3.

For queued request source 3, the Trigger Control Register configures the trigger sequence counter and the internal trigger sources.

GxTRCTR (x = 0 - 3)
Trigger Control Register, Group x

($x * 0400_H + 0554_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COV	0	0	SR DIS	0	0	IT SEL		0	0	TSCSET					
w	r	r	rw	r	r	rw		r	r	rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OV	Q3 ACT	0	0	0	0	0	0	0	0	TSC					
rh	rh	r	r	r	r	r	r	r	r	rh					

Field	Bits	Type	Description
TSC	[5:0]	rh	**Trigger Sequence Counter** Controls the effect of an incoming internal trigger: If TSC = $00\ 0000_B$: start the sequence in queue 3 and reload TSC from bitfield TSCSET.[1)] If TSC > $00\ 0000_B$: decrement TSC by one. 00_H Issue conversion requests immediately others: decrement trigger counter
0	[13:6]	r	**Reserved, write 0, read as 0**
Q3ACT	14	rh	**Queue 3 Active** Indicates that request source 3 is currently executing a sequence. 0_B No activity 1_B Queue 3 currently active *Note: Cleared by writing 1 to bit COV.*
OV	15	rh	**Overflow Detected** Indicates that a trigger has been activated while the queue was still active (Q3ACT = 1). 0_B No trigger event 1_B An irregular trigger event has been detected *Note: Cleared by writing 1 to bit COV.*

Field	Bits	Type	Description
TSCSET	[21:16]	rw	**Trigger Sequence Counter Start Value** Defines the initial value of the trigger sequence counter TSC. TSC is reloaded with the value in TSCSET, when a trigger occurs while TSC = 00_H. TSCSET is automatically copied to TSC when being written.[1)]
0	[23:22]	r	**Reserved, write 0, read as 0**
ITSEL	[25:24]	rw	**Internal Trigger Input Selection** Internal triggers are generated by the respective source events. Enable a source event in the selected request source to generate an internal trigger signal. 00_B Select queued request source (0) 01_B Select scan request source (1) 10_B Select background source (2) 11_B Reserved *Note: The selected trigger signal is internally connected to gate input GxREQGTP of this request source. It is selected when XTSEL = GTSEL = 1111_H.*
0	[27:26]	r	**Reserved, write 0, read as 0**
SRDIS	28	rw	**Service Request Disable** Controls if the source event of the selected trigger source also activates a service request. 0_B Source event generates service request 1_B No service request, only internal trigger generated
0	[30:29]	r	**Reserved, write 0, read as 0**
COV	31	w	**Clear Overflow Flag** 0_B No action 1_B Clear bits OV and Q3ACT

1) Write zero to bitfield TSCSET, to disable the trigger sequence counter and start the sequence in queue 3 with each incoming trigger.

*Note: For more information, please see **Section 27.11.4**.*

27.5.2 Channel Scan Request Source Handling

The VADC provides two types of channel scan sources:

- Source 1: Group scan source
  This scan source can request all channels of the corresponding group.
- Source 2: Background scan source
  This scan source can request all channels of all groups.
  Priority channels selected in registers **GxCHASS (x = 0 - 3)** cannot take part in background conversion sequences.

Both sources operate in the same way and provide the same register interface. The background source provides more request/pending bits because it can request all channels of all groups.

Each analog input channel can be included in or excluded from the scan sequence by setting or clearing the corresponding channel select bit in register **GxASSEL (x = 0 - 3)** or **BRSSELx (x = 0 - 3)**. The programmed register value remains unchanged by an ongoing scan sequence. The scan sequence starts with the highest enabled channel number and continues towards lower channel numbers.

Upon a load event, the request pattern is transferred to the pending bits in register **GxASPND (x = 0 - 3)** or **BRSPNDx (x = 0 - 3)**. The pending conversion requests indicate which input channels are to be converted in an ongoing scan sequence. Each conversion start that was triggered by the scan request source, automatically clears the corresponding pending bit. If the last conversion triggered by the scan source is finished and all pending bits are cleared, the current scan sequence is considered finished and a request source event is generated.

A conversion request is only issued to the request source arbiter if at least one pending bit is set.

If the arbiter aborts a conversion triggered by the scan request source due to higher priority requests, the corresponding pending bit is automatically set. This ensures that an aborted conversion is not lost but takes part in the next arbitration round.

The trigger and gating unit generates load events from the selected external (outside the ADC) trigger and gating signals. For example, a timer unit can issue a request signal to synchronize conversions to PWM events.

Load events start a scan sequence and can be generated either via software or via the selected hardware triggers. The request source event can also generate an automatic load event, so the programmed sequence is automatically repeated.

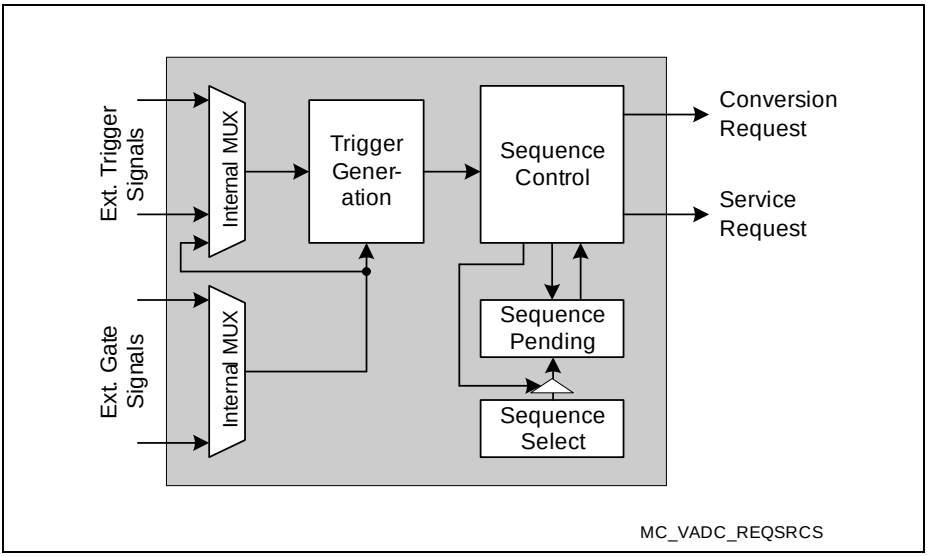


Figure 27-10 Scan Request Source

Scan Source Operation

Configure the scan request source by executing the following actions:

- Select the input channels for the sequence by programming **GxASSEL (x = 0 - 3)** or **BRSSELx (x = 0 - 3)**
- If hardware trigger or gating is desired, select the appropriate trigger and gating inputs and the proper signal transitions by programming **GxASCTRL (x = 0 - 3)** or **BRSCTRL**. Enable the trigger and select the gating mode by programming **GxASMR (x = 0 - 3)** or **BRSMR**.[1)]
- Define the load event operation (handling of pending bits, autoscan mode) by programming **GxASMR (x = 0 - 3)** or **BRSMR**.
  A load event with bit LDM = 0 copies the content of **GxASSEL (x = 0 - 3)** or **BRSSELx (x = 0 - 3)** to **GxASPND (x = 0 - 3)** or **BRSPNDx (x = 0 - 3)** (overwrite mode). This starts a new scan sequence and aborts any pending conversions from a previous scan sequence.
  A load event with bit LDM = 1 OR-combines the content of **GxASSEL (x = 0 - 3)** or **BRSSELx (x = 0 - 3)** to **GxASPND (x = 0 - 3)** or **BRSPNDx (x = 0 - 3)** (combine mode). This starts a scan sequence that includes pending conversions from a prvious scan sequence.
- Enable the corresponding arbitration slot (1) to accept conversion requests from the channel scan source (see register **GxARBPR (x = 0 - 3)**).

Start a channel scan sequence by generating a load event:

- If a hardware trigger is selected and enabled, generate the configured transition at the selected input signal, e.g. from a timer or an input pin.
- Generate a software load event by setting LDEV = 1 (**GxASMR (x = 0 - 3)** or **BRSMR**).
- Generate a load event by writing the scan pattern directly to the pending bits in **GxASPND (x = 0 - 3)** or **BRSPNDx (x = 0 - 3)**. The pattern is copied to **GxASSEL (x = 0 - 3)** or **BRSSELx (x = 0 - 3)** and a load event is generated automatically.
  In this case, a scan sequence can be defined and started with a single data write action, e.g. under PEC control (provided that the pattern fits into one register).

Note: If autoscan is enabled, a load event is generated automatically each time a request source event occurs when the scan sequence has finished. This permanently repeats the defined scan sequence (autoscan).

Stop or abort an ongoing scan sequence by executing the following actions:

- If external gating is enabled, switch the gating signal to the defined inactive level. This does not modify the conversion pending bits, but only prevents issuing conversion requests to the arbiter.

1) To avoid unexpected signal transitions, select/enable the trigger and/or gate inputs before activating the corresponding request source. If PDOUT signals from the ERU are used, first initialize the ERU.

- Disable the corresponding arbitration slot (1 or 2) in the arbiter. This does not modify the contents of the conversion pending bits, but only prevents the arbiter from accepting requests from the request handling block.
- Disable the channel scan source by clearing bitfield ENGT = 00_B. Clear the pending request bits by setting bit CLRPND = 1 (**GxASMR (x = 0 - 3)** or **BRSMR**).

Scan Request Source Events and Service Requests

A request source event of a scan source occurs if the last conversion of a scan sequence is finished (all pending bits = 0). A request source event interrupt can be generated based on a request source event. If a request source event is detected, it sets the corresponding indication flag in register **GxSEFLAG (x = 0 - 3)**. These flags can also be set by writing a 1 to the corresponding bit position, whereas writing 0 has no effect.

The service request output SRx that is selected by the request source event interrupt node pointer bitfields in register **GxSEVNP (x = 0 - 3)** becomes activated each time the related request source event is detected (and enabled by ENSI) or the related bit position in register **GxSEFLAG (x = 0 - 3)** is written with a 1 (this write action simulates a request source event).

The indication flags can be cleared by SW by writing a 1 to the corresponding bit position in register **GxSEFCLR (x = 0 - 3)**.[1)]

1) Please refer to **"Service Request Generation" on Page 27-153**.

Registers of Group Scan Source

There is a separate register set for each group scan source. These sources can be operated independently.

The control register of the autoscan source selects the external gate and/or trigger signals.
Write control bits allow separate control of each function with a simple write access.

GxASCTRL (x = 0 - 3)
Autoscan Source Control Register, Group x
(x * 0400_H + 0520_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TM WC	0	0	TM EN	0	0	0	0	GT WC	0	0	GT LVL	GT SEL			
w	r	r	rw	r	r	r	r	w	r	r	rh	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XT WC	XT MODE		XT LVL	XT SEL				0	0	0	0	SRCRESREG			
w	rw		rh	rw				r	r	r	r	rw			

Field	Bits	Type	Description
SRCRESREG	[3:0]	rw	**Source-specific Result Register** 0000_B Use GxCHCTRy.RESREG to select a group result register 0001_B Store result in group result register GxRES1 … 1111_B Store result in group result register GxRES15
0	[7:4]	r	**Reserved, write 0, read as 0**
XTSEL	[11:8]	rw	**External Trigger Input Selection** The connected trigger input signals are listed in **Table 27-13 "Digital Connections in the TC21x/TC22x/TC23x" on Page 27-176** *Note: XTSEL = 1111_B uses the selected gate input as trigger source (ENGT must be $0X_B$).*
XTLVL	12	rh	**External Trigger Level** Current level of the selected trigger input

Field	Bits	Type	Description
XTMODE	[14:13]	rw	**Trigger Operating Mode** 00_B No external trigger 01_B Trigger event upon a falling edge 10_B Trigger event upon a rising edge 11_B Trigger event upon any edge
XTWC	15	w	**Write Control for Trigger Configuration** 0_B No write access to trigger configuration 1_B Bitfields XTMODE and XTSEL can be written
GTSEL	[19:16]	rw	**Gate Input Selection** The connected gate input signals are listed in **Table 27-13 “Digital Connections in the TC21x/TC22x/TC23x” on Page 27-176**
GTLVL	20	rh	**Gate Input Level** Current level of the selected gate input
0	[22:21]	r	**Reserved, write 0, read as 0**
GTWC	23	w	**Write Control for Gate Configuration** 0_B No write access to gate configuration 1_B Bitfield GTSEL can be written
0	[27:24]	r	**Reserved, write 0, read as 0**
TMEN	28	rw	**Timer Mode Enable** 0_B No timer mode: standard gating mechanism can be used 1_B Timer mode for equidistant sampling enabled: standard gating mechanism must be disabled
0	[30:29]	r	**Reserved, write 0, read as 0**
TMWC	31	w	**Write Control for Timer Mode** 0_B No write access to timer mode 1_B Bitfield TMEN can be written

The Conversion Request Mode Register configures the operating mode of the channel scan request source.

GxASMR (x = 0 - 3)
Autoscan Source Mode Register, Group x
(x * 0400_H + 0524_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RPT DIS
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	LD EV	CLR PND	REQ GT	0	LDM	SCA N	EN SI	EN TR	ENGT	
r	r	r	r	r	r	w	w	rh	r	rw	rw	rw	rw	rw	

Field	Bits	Type	Description
ENGT	[1:0]	rw	**Enable Gate** Selects the gating functionality for source 1. 00_B No conversion requests are issued 01_B Conversion requests are issued if at least one pending bit is set 10_B Conversion requests are issued if at least one pending bit is set and REQGTx = 1. 11_B Conversion requests are issued if at least one pending bit is set and REQGTx = 0. *Note: REQGTx is the selected gating signal.*
ENTR	2	rw	**Enable External Trigger** 0_B External trigger disabled 1_B The selected edge at the selected trigger input signal REQTR generates the load event
ENSI	3	rw	**Enable Source Interrupt** 0_B No request source interrupt 1_B A request source interrupt is generated upon a request source event (last pending conversion is finished)

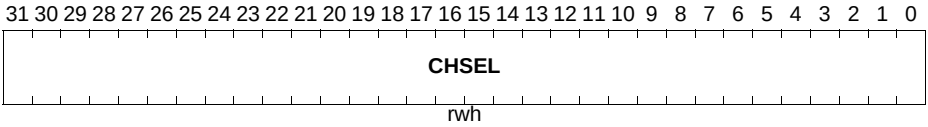
Field	Bits	Type	Description
SCAN	4	rw	**Autoscan Enable** 0_B No autoscan 1_B Autoscan functionality enabled: a request source event automatically generates a load event
LDM	5	rw	**Autoscan Source Load Event Mode** 0_B Overwrite mode: Copy all bits from the select registers to the pending registers upon a load event 1_B Combine mode: Set all pending bits that are set in the select registers upon a load event (logic OR)
0	6	r	**Reserved, write 0, read as 0**
REQGT	7	rh	**Request Gate Level** Monitors the level at the selected REQGT input. 0_B The gate input is low 1_B The gate input is high
CLRPND	8	w	**Clear Pending Bits** 0_B No action 1_B The bits in register GxASPNDx are cleared
LDEV	9	w	**Generate Load Event** 0_B No action 1_B A load event is generated
0	[15:10]	r	**Reserved, write 0, read as 0**
RPTDIS	16	rw	**Repeat Disable** 0_B A cancelled conversion is repeated 1_B A cancelled conversion is discarded
0	[31:17]	r	**Reserved, write 0, read as 0**

The Channel Select Register selects the channels to be converted by the group scan request source. Its bits are used to update the pending register, when a load event occurs.

The number of valid channel bits depends on the channels available in the respective product type (please refer to **“Product-Specific Configuration” on Page 27-167**).

GxASSEL (x = 0 - 3)
Autoscan Source Channel Select Register, Group x

($x * 0400_H + 0528_H$) **Reset Value: $0000\ 0000_H$**



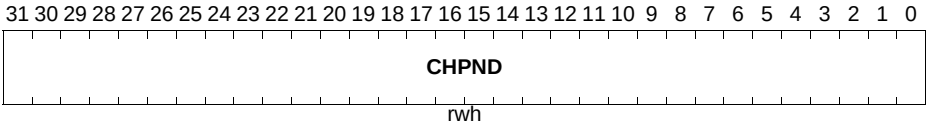
Field	Bits	Type	Description
CHSEL	[31:0]	rwh	**Channel Selection** Each bit (when set) enables the corresponding input channel of the respective group to take part in the scan sequence. 0_B Ignore this channel 1_B This channel is part of the scan sequence

Note: Register GxASSEL is also updated when writing a pattern to register GxASPND.

The Channel Pending Register indicates the channels to be converted in the current conversion sequence. They are updated from the select register, when a load event occurs.

GxASPND (x = 0 - 3)
Autoscan Source Pending Register, Group x

($x * 0400_H + 052C_H$) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CHPND	[31:0]	rwh	**Channels Pending** Each bit (when set) request the conversion of the corresponding input channel of the respective group. 0_B Ignore this channel 1_B Request conversion of this channel

Note: Writing to register GxASPND automatically updates register GxASSEL.

Registers of Background Scan Source

There is a single register set for the background scan source. This source is common for the complete VADC.

The control register of the background request source selects the external gate and/or trigger signals.
Write control bits allow separate control of each function with a simple write access.

BRSCTRL
Background Request Source Control Register
(0200_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	GT WC	0	0	GT LVL	GT SEL			
r	r	r	r	r	r	r	r	w	r	r	rh	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XT WC	XT MODE		XT LVL	XT SEL				0	0	0	0	SRCRESREG			
w	rw		rh	rw				r	r	r	r	rw			

Field	Bits	Type	Description
SRCRESREG	[3:0]	rw	**Source-specific Result Register** 0000_B Use GxCHCTRy.RESREG to select a group result register 0001_B Store result in group result register GxRES1 … 1111_B Store result in group result register GxRES15
0	[7:4]	r	**Reserved, write 0, read as 0**
XTSEL	[11:8]	rw	**External Trigger Input Selection** The connected trigger input signals are listed in **Table 27-13 “Digital Connections in the TC21x/TC22x/TC23x” on Page 27-176** *Note: XTSEL = 1111_B uses the selected gate input as trigger source (ENGT must be $0X_B$).*
XTLVL	12	rh	**External Trigger Level** Current level of the selected trigger input

Field	Bits	Type	Description
XTMODE	[14:13]	rw	**Trigger Operating Mode** 00_B No external trigger 01_B Trigger event upon a falling edge 10_B Trigger event upon a rising edge 11_B Trigger event upon any edge
XTWC	15	w	**Write Control for Trigger Configuration** 0_B No write access to trigger configuration 1_B Bitfields XTMODE and XTSEL can be written
GTSEL	[19:16]	rw	**Gate Input Selection** The connected gate input signals are listed in **Table 27-13 “Digital Connections in the TC21x/TC22x/TC23x” on Page 27-176**
GTLVL	20	rh	**Gate Input Level** Current level of the selected gate input
0	[22:21]	r	**Reserved, write 0, read as 0**
GTWC	23	w	**Write Control for Gate Configuration** 0_B No write access to gate configuration 1_B Bitfield GTSEL can be written
0	[31:24]	r	**Reserved, write 0, read as 0**

The Conversion Request Mode Register configures the operating mode of the background request source.

BRSMR
Background Request Source Mode Register
(0204_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RPT DIS
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	LD EV	CLR PND	REQ GT	0	LDM	SCA N	EN SI	EN TR	ENGT	
r	r	r	r	r	r	w	w	rh	r	rw	rw	rw	rw	rw	

Field	Bits	Type	Description
ENGT	[1:0]	rw	**Enable Gate** Selects the gating functionality for source 1. 00_B No conversion requests are issued 01_B Conversion requests are issued if at least one pending bit is set 10_B Conversion requests are issued if at least one pending bit is set and REQGTx = 1. 11_B Conversion requests are issued if at least one pending bit is set and REQGTx = 0. *Note: REQGTx is the selected gating signal.*
ENTR	2	rw	**Enable External Trigger** 0_B External trigger disabled 1_B The selected edge at the selected trigger input signal REQTR generates the load event
ENSI	3	rw	**Enable Source Interrupt** 0_B No request source interrupt 1_B A request source interrupt is generated upon a request source event (last pending conversion is finished)

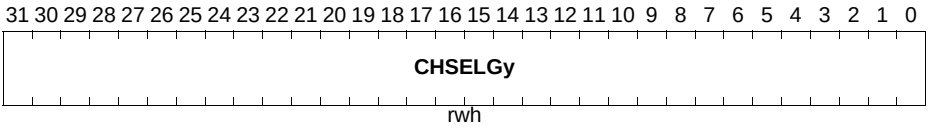
Field	Bits	Type	Description
SCAN	4	rw	**Autoscan Enable** 0_B No autoscan 1_B Autoscan functionality enabled: a request source event automatically generates a load event
LDM	5	rw	**Autoscan Source Load Event Mode** 0_B Overwrite mode: Copy all bits from the select registers to the pending registers upon a load event 1_B Combine mode: Set all pending bits that are set in the select registers upon a load event (logic OR)
0	6	r	**Reserved, write 0, read as 0**
REQGT	7	rh	**Request Gate Level** Monitors the level at the selected REQGT input. 0_B The gate input is low 1_B The gate input is high
CLRPND	8	w	**Clear Pending Bits** 0_B No action 1_B The bits in registers BRSPNDx are cleared
LDEV	9	w	**Generate Load Event** 0_B No action 1_B A load event is generated
0	[15:10]	r	**Reserved, write 0, read as 0**
RPTDIS	16	rw	**Repeat Disable** 0_B A cancelled conversion is repeated 1_B A cancelled conversion is discarded
0	[31:17]	r	**Reserved, write 0, read as 0**

The Channel Select Registers select the channels to be converted by the background request source (channel scan source). Its bits are used to update the pending registers, when a load event occurs.

The number of valid channel bits depends on the channels available in the respective product type (please refer to **"Product-Specific Configuration" on Page 27-167**).

*Note: Priority channels selected in registers **GxCHASS (x = 0 - 3)** will not be converted.*

BRSSELx (x = 0 - 3)
Background Request Source Channel Select Register, Group x
(0180_H + x * 0004_H) **Reset Value: 0000 0000_H**

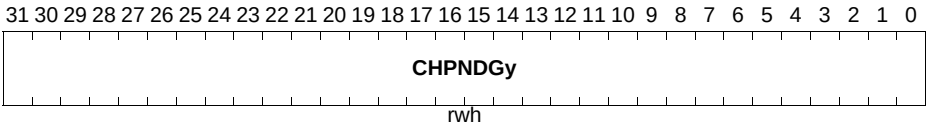


Field	Bits	Type	Description
CHSELGy	[31:0]	rwh	**Channel Selection Group x** Each bit (when set) enables the corresponding input channel of the respective group to take part in the background scan sequence. 0_B Ignore this channel 1_B This channel is part of the scan sequence

The Channel Pending Registers indicate the channels to be converted in the current conversion sequence. They are updated from the select registers, when a load event occurs.

BRSPNDx (x = 0 - 3)
Background Request Source Pending Register, Group x

($01C0_H$ + x * 0004_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CHPNDGy	[31:0]	rwh	**Channels Pending Group x** Each bit (when set) request the conversion of the corresponding input channel of the respective group. 0_B Ignore this channel 1_B Request conversion of this channel

Note: Writing to any of registers BRSPNDx automatically updates the corresponding register BRSSELx and generates a load event that copies all bits from all registers BRSSELx to BRSPNDx.
Use this shortcut only when writing the last word of the request pattern.

27.6 Request Source Arbitration

The request source arbiter regularly polls the request sources, one after the other, for pending conversion requests. Each request source is assigned to a certain time slot within an arbitration round, called arbitration slot. The duration of an arbitration slot is user-configurable via register GLOBCFG, see **Global Configuration**.

The priority of each request source is user-configurable via register **GxARBPR (x = 0 - 3)**, so the arbiter can select the next channel to be converted, in the case of concurrent requests from multiple sources, according to the application requirements.

An unused arbitration slot is considered empty and does not take part in the arbitration. After reset, all slots are disabled and must be enabled (register **GxARBPR (x = 0 - 3)**) to take part in the arbitration process.

Figure 27-11 summarizes the arbitration sequence. An arbitration round consists of one arbitration slot for each available request source. The synchronization source is always evaluated in the last slot and has a higher priority than all other sources. At the end of each arbitration round, the arbiter has determined the highest priority conversion request.

If a conversion is started in an arbitration round, this arbitration round does not deliver an arbitration winner. In the TC21x/TC22x/TC23x, the following request sources are available:

- Arbitration slot 0: **Group Queued source**, 8-stage sequences in arbitrary order
- Arbitration slot 1: **Group Scan source**, sequences in defined order within group
- Arbitration slot 2: **Background Scan source**, sequences in defined order, all groups
- Arbitration slot 3: **Group Queued source**, 8-stage sequences in arbitrary order, automatic control of test features
- Last arbitration slot: **Synchronization source**, synchronized conversion requests from another ADC kernel (always handled with the highest priority in a synchronization slave kernel).

Note: If request source 3 is not used, the shortest possible arbitration round of 4 arbitration slots (ARBRND = 00_B) can be selected. This provides the fastest arbiter operation.
To use request source 3, the arbitration round must be increased to 8 slots (ARBRND = 01_B). In this case, slots 0 … 3 are used for sources 0 … 3 and slot 7 is the synchronization slot (slots 4 … 6 are unused).
***Figure 27-11** shows the shortest possible arbitration round of 4 slots.*

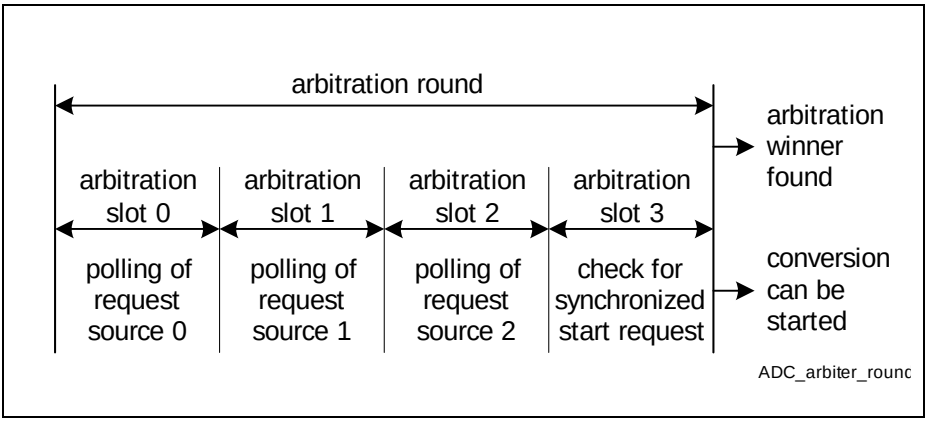


Figure 27-11 Arbitration Round with 4 Arbitration Slots

27.6.1 Arbiter Operation and Configuration

The timing of the arbiter (i.e. of an arbitration round) is determined by the number of arbitration slots within an arbitration round and by the duration of an arbitration slot.

An arbitration round consist of 4…20 arbitration slots (defined by bitfield **GxARBCFG (x = 0 - 3)**.ARBRND). 4 or 8 slots are sufficient for the TC21x/TC22x/TC23x (depending on the usage of source 3), more can be programmed to obtain the same arbiter timing for different products.

The duration of an arbitration slot is configurable t_{Slot} = (DIVD+1) / f_{ADC}.
The duration of an arbitration round, therefore, is t_{ARB} = 4 × t_{Slot} (or 8 × t_{Slot}).

The period of the arbitration round introduces a timing granularity to detect an incoming conversion request signal and the earliest point to start the related conversion. This granularity can introduce a jitter of maximum one arbitration round. The jitter can be reduced by minimizing the period of an arbitration round.

To achieve a reproducible reaction time (constant delay without jitter) between the trigger event of a conversion request (e.g. by a timer unit or due to an external event) and the start of the related conversion, mainly the following two options exist. For both options, the converter has to be idle and other conversion requests must not be pending for at least one arbiter round before the trigger event occurs:

- If bit **GxARBCFG (x = 0 - 3)**.ARBM = 0, the **arbiter runs permanently**. In this mode, synchronized conversions of more than one ADC kernel are possible.[1)]
  The trigger for a conversion request has to be generated synchronously to the arbiter timing. Incoming triggers should have exactly n-times the granularity of the arbiter (n = 1, 2, 3,...). In order to allow some flexibility, the duration of an arbitration slot can be programmed in cycles of f_{ADC}.
- If bit **GxARBCFG (x = 0 - 3)**.ARBM = 1, the **arbiter stops after an arbitration round** when no conversion request have been found pending any more. The arbiter is started again if at least one enabled request source indicates a pending conversion request. The trigger for a conversion request does not need to be synchronous to the arbiter timing.
  In this mode, parallel conversions are not possible for synchronization slave kernels.

Each request source has a configurable priority, so the arbiter can resolve concurrent conversion requests from different sources. The request with the highest priority is selected for conversion. These priorities can be adapted to the requirements of a given application (see register **GxARBPR (x = 0 - 3)**).

The **Conversion Start Mode** determines the handling of the conversion request that has won the arbitration.

1) For more information, please refer to **"Synchronization of Conversions" on Page 27-132**.

The Arbitration Configuration Register selects the timing and the behavior of the arbiter.

GxARBCFG (x = 0 - 3)
Arbitration Configuration Register, Group x
($x * 0400_H + 0480_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SAMPLE	BUSY	CALS	CAL	0	0	SYNRUN	CHNR					CSRC		ANONS	
rh	rh	rh	rh	r	r	rh	rh					rh		rh	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	ARBM	0	ARBRND		0	0	ANONC	
r	r	r	r	r	r	r	r	rw	r	rw		r	r	rw	

Field	Bits	Type	Description
ANONC	[1:0]	rw	**Analog Converter Control** Defines the value of bitfield ANONS in a stand-alone converter or a converter in master mode. Coding see ANONS or **Section 27.4.1**.
0	[3:2]	r	**Reserved, write 0, read as 0**
ARBRND	[5:4]	rw	**Arbitration Round Length** Defines the number of arbitration slots per arb. round (arbitration round length = t_{ARB}).[1)] 00_B 4 arbitration slots per round ($t_{ARB} = 4 / f_{ADCD}$) 01_B 8 arbitration slots per round ($t_{ARB} = 8 / f_{ADCD}$) 10_B 16 arbitration slots per round ($t_{ARB} = 16 / f_{ADCD}$) 11_B 20 arbitration slots per round ($t_{ARB} = 20 / f_{ADCD}$)
0	6	r	**Reserved, write 0, read as 0**

Field	Bits	Type	Description
ARBM	7	rw	**Arbitration Mode** 0_B The arbiter runs permanently. This setting is required for a synchronization slave (see **Section 27.10.1**) and for equidistant sampling using the signal ARBCNT (see **Section 27.10.2**). 1_B The arbiter only runs if at least one conversion request of an enabled request source is pending. This setting ensures a reproducible latency from an incoming request to the conversion start, if the converter is idle. Synchronized conversions are not supported.
0	[15:8]	r	**Reserved, write 0, read as 0**
ANONS	[17:16]	rh	**Analog Converter Control Status** Defined by bitfield ANONC in a stand-alone kernel or a kernel in master mode. In slave mode, this bitfield is defined by bitfield ANONC of the respective master kernel. See also **Section 27.4.1**. 00_B Analog converter off 01_B Reserved 10_B Reserved 11_B Normal operation (permanently on)
CSRC	[19:18]	rh	**Currently Converted Request Source** Indicates the arbitration slot number of the current (BUSY = 1) or of the last (BUSY = 0) conversion. This bitfield is updated when a conversion is started. 00_B Current/last conversion for request source 0 01_B Current/last conversion for request source 1 10_B Current/last conversion for background source 11_B Current/last conversion for request source 3 or synchronization request (slave converter) [2)]
CHNR	[24:20]	rh	**Channel Number** Indicates the current or last converted analog input channel. This bitfield is updated when a conversion is started.

Field	Bits	Type	Description
SYNRUN	25	rh	**Synchronous Conversion Running** Indicates that a synchronized (= parallel) conversion is currently running. 0_B Normal conversion or no conversion running 1_B A synchronized conversion is running (cannot be cancelled by higher priority requests!)
0	[27:26]	r	**Reserved, write 0, read as 0**
CAL	28	rh	**Start-Up Calibration Active Indication** Indicates the start-up calibration phase of the corresponding analog converter. 0_B Completed or not yet started 1_B Start-up calibration phase is active (set one clock cycle after setting bit SUCAL) *Note: Start conversions only after the start-up calibration phase is complete.*[3]
CALS	29	rh	**Start-Up Calibration Started** Indicates that the start-up calibration has begun. 0_B Requested but not yet started 1_B Start-up calibration has begun *Note: Bit CALS is cleared when setting bit SUCAL and is set together with bit CAL.*
BUSY	30	rh	**Converter Busy Flag** 0_B Not busy 1_B Converter is busy with a conversion
SAMPLE	31	rh	**Sample Phase Flag** 0_B Converting or idle 1_B Input signal is currently sampled

1) The default setting of 4 arbitration slots is sufficient for correct arbitration of request sources 0 … 2. To include request source 3 select 8 arbitration slots. The duration of an arbitration round can be increased if required to synchronize requests.

2) If request source 3 is available/active and/or synchronized conversions are configured.

3) The completion of the start-up calibration is indicated by CAL = 0 AND CALS = 1.
CAL = CALS = 0 indicates that the start-up calibration was requested but has not yet begun.
CAL = CALS = 1 indicates an ongoing calibration phase.

The Arbitration Priority Register defines the request source priority and the conversion start mode for each request source.

Note: Only change priority and conversion start mode settings of a request source while this request source is disabled, and a currently running conversion requested by this source is finished.

GxARBPR (x = 0 - 3)
Arbitration Priority Register, Group x

(x * 0400_H + 0484_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	ASEN3	ASEN2	ASEN1	ASEN0	0	0	0	0	0	0	0	0
r	r	r	r	rw	rw	rw	rw	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSM3	0	PRIO3		CSM2	0	PRIO2		CSM1	0	PRIO1		CSM0	0	PRIO0	
rw	r	rw		rw	r	rw		rw	r	rw		rw	r	rw	

Field	Bits	Type	Description
PRIO0, PRIO1, PRIO2, PRIO3	[1:0], [5:4], [9:8], [13:12]	rw	**Priority of Request Source x** Arbitration priority of request source x (in slot x) 00_B Lowest priority is selected. ... 11_B Highest priority is selected.
CSM0, CSM1, CSM2, CSM3	3, 7, 11, 15	rw	**Conversion Start Mode of Request Source x** 0_B Wait-for-start mode 1_B Cancel-inject-repeat mode, i.e. this source can cancel conversion of other sources.
0	2, 6, 10, 14, [23:16]	r	**Reserved, write 0, read as 0**

Field	Bits	Type	Description
ASENy (y = 0 - 3)	24 + y	rw	**Arbitration Slot y Enable** Enables the associated arbitration slot of an arbiter round. The request source bits are not modified by write actions to ASENR. 0_B The corresponding arbitration slot is disabled and considered as empty. Pending conversion requests from the associated request source are disregarded. 1_B The corresponding arbitration slot is enabled. Pending conversion requests from the associated request source are arbitrated.
0	[31:28]	r	**Reserved, write 0, read as 0**

27.6.2 Conversion Start Mode

When the arbiter has selected the request to be converted next, the handling of this channel depends on the current activity of the converter:

- Converter is currently idle: the conversion of the arbitration winner is started immediately.
- Current conversion has same or higher priority: the current conversion is completed, the conversion of the arbitration winner is started after that.
- Current conversion has lower priority: the action is user-configurable:
  - **Wait-for-start mode**: the current conversion is completed, the conversion of the arbitration winner is started after that. This mode provides maximum throughput, but can produce a jitter for the higher priority conversion.
    Example in **Figure 27-12**:
    Conversion A is requested (t1) and started (t2). Conversion B is then requested (t3), but started only after completion of conversion A (t4).
  - **Cancel-inject-repeat mode**: the current conversion is aborted, the conversion of the arbitration winner is started after the abortion (3 f_{ADC} cycles).
    The aborted conversion request is restored in the corresponding request source and takes part again in the next arbitration round. This mode provides minimum jitter for the higher priority conversions, but reduces the overall throughput.
    Example in **Figure 27-12**:
    Conversion A is requested (t6) and started (t7). Conversion B is then requested (t8) and started (t9), while conversion A is aborted but requested again. When conversion B is complete (t10), conversion A is restarted.
    Exception: If both requests target the same result register with wait-for-read mode active (see **Section 27.9.3**), the current conversion cannot be aborted.

Note: A cancelled conversion can be repeated automatically in each case, or it can be discarded if it was cancelled. This is selected for each source by bit RPTDIS in the corresponding source's mode register.

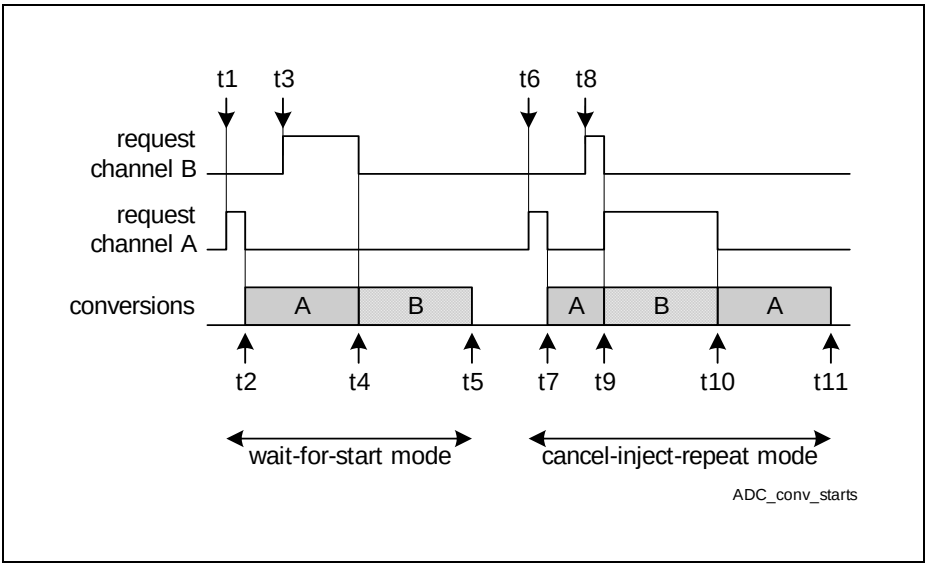


Figure 27-12 Conversion Start Modes

The conversion start mode can be individually programmed for each request source by bits in register **GxARBPR (x = 0 - 3)** and is applied to all channels requested by the source. In this example, channel A is issued by a request source with a lower priority than the request source requesting the conversion of channel B.

27.7 Analog Input Channel Configuration

For each analog input channel a number of parameters can be configured that control the conversion of this channel. The channel control registers define the following parameters:

- **Channel Parameters**: The sample time for this channel and the data width of the result are defined via input classes. Each channel can select one of two classes of its own group or one of two global classes.
- **Reference selection**: an alternate reference voltage can be selected for most channels (exceptions are marked in **Section 27.14.3**)
- **Result target**: The conversion result values are stored either in a group-specific result register or in the global result register. The group-specific result registers are selected in several ways:
  - channel-specific, selected by bitfield RESREG in register **G0CHCTRy (y = 0 - 11)** etc., with bitfield SRCRESREG = 0000_B
  - source-specific, selected by bitfield SRCRESREG in register **GxQCTRL0 (x = 0 - 3)**, **GxASCTRL (x = 0 - 3)** or **BRSCTRL** with bitfield SRCRESREG ≠ 0000_B
- **Result position**: The result values can be stored left-aligned or right-aligned. The exact position depends also on the configured result width and on the data accumulation mode.
  See also **Figure 27-19 “Result Storage Options” on Page 27-121**.
- **Compare with Standard Conversions (Limit Checking)**: Channel events can be generated whenever a new result value becomes available. Channel event generation can be restricted to values that lie inside or outside a user-configurable band.
  In Fast Compare Mode, channel events can be generated depending on the transitions of the (1-bit) result.
- **Broken Wire Detection**: This safety feature can detect a missing connection to an analog signal source (sensor).
- **Synchronization of Conversions**: Synchronized conversions are executed at the same time on several converters.

The **Alias Feature** redirects conversion requests for channels CH0 and/or CH1 to other channels.

27.7.1 Channel Parameters

Each analog input channel is configured by its associated channel control register.

*Note: For the safety feature “Broken Wire Detection”, refer to **Section 27.11.1**.*

The following features can be defined for each channel:

- The conversion class defines the result width and the sample time
- Generation of channel events and the result value band, if used
- Target of the result defining the target register and the position within the register

G0CHCTRy (y = 0 - 11)
Group 0, Channel y Ctrl. Reg. (0600_H + y * 0004_H) Reset Value: 0000 0000_H
G1CHCTRy (y = 0 - 11)
Group 1, Channel y Ctrl. Reg. ($0A00_H$ + y * 0004_H) Reset Value: 0000 0000_H
G2CHCTR0
Group 2, Channel 0 Ctrl. Reg. ($0E00_H$) Reset Value: 0000 0000_H
G3CHCTR0
Group 3, Channel 0 Ctrl. Reg. (1200_H) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	BWDEN	BWDCH		0	0	0	0	0	0	RESPOS	RESTBS	RESREG			
r	rw	rw		r	r	r	r	r	r	rw	rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BNDSELX				REFSEL	SYNC	CHEVMODE		BNDSELU		BNDSELL		0	0	ICLSEL	
rw				rw	rw	rw		rw		rw		r	r	rw	

Field	Bits	Type	Description
ICLSEL	[1:0]	rw	**Input Class Select** 00_B Use group-specific class 0 01_B Use group-specific class 1 10_B Use global class 0 11_B Use global class 1
0	[3:2]	r	**Reserved, write 0, read as 0**
BNDSELL	[5:4]	rw	**Lower Boundary Select**[1)] 00_B Use group-specific boundary 0 01_B Use group-specific boundary 1 10_B Use global boundary 0 11_B Use global boundary 1
BNDSELU	[7:6]	rw	**Upper Boundary Select**[1)] 00_B Use group-specific boundary 0 01_B Use group-specific boundary 1 10_B Use global boundary 0 11_B Use global boundary 1

Field	Bits	Type	Description
CHEVMODE	[9:8]	rw	**Channel Event Mode** Generate a channel event either in normal compare mode (NCM) with limit checking[2)] or in Fast Compare Mode (FCM)[3)] 00_B Never 01_B NCM: If result is inside the boundary band FCM: If result becomes high (above cmp. val.) 10_B NCM: If result is outside the boundary band FCM: If result becomes low (below cmp. val.) 11_B NCM: Always (ignore band) FCM: If result switches to either level
SYNC	10	rw	**Synchronization Request** 0_B No synchroniz. request, standalone operation 1_B Request a synchronized conversion of this channel (only taken into account for a master)
REFSEL	11	rw	**Reference Input Selection** Defines the reference voltage input to be used for conversions on this channel. 0_B Standard reference input V_{AREF} 1_B Alternate reference input from CH0[4)]
BNDSELX	[15:12]	rw	**BoundaryExtension[1)]** 0000_B Standard mode: select boundaries via BNDSELU/BNDSELL 0001_B Use result reg. GxRES1 as upper boundary … 1111_B Use result reg. GxRES15 as upper boundary
RESREG	[19:16]	rw	**Result Register** 0000_B Store result in group result register GxRES0 … 1111_B Store result in group result register GxRES15
RESTBS	20	rw	**Result Target for Background Source** 0_B Store results in the selected group result register 1_B Store results in the global result register
RESPOS	21	rw	**Result Position** 0_B Store results left-aligned 1_B Store results right-aligned
0	[27:22]	r	**Reserved, write 0, read as 0**

Field	Bits	Type	Description
BWDCH	[29:28]	rw	**Broken Wire Detection Channel** 00_B Select V_{AREF} 01_B Select V_{AGND} 10_B Reserved 11_B Reserved
BWDEN	30	rw	**Broken Wire Detection Enable** 0_B Normal operation 1_B Additional preparation phase is enabled
0	31	r	**Reserved, write 0, read as 0**

1) While BNDSELX ≠ 0000_B, bitfields BNDSELU and BNDSELL are concatenated and select the corresponding result register as lower boundary.

2) The boundary band is defined as the area where the result is less than or equal to the selected upper boundary and greater than or equal to the selected lower boundary, see **Section 27.7.4**.

3) The result is bit FCR in the selected result register.

4) Some channels cannot select an alternate reference.

Input Class Registers

The group-specific input class registers define the sample time and data conversion mode for each channel of the respective group that selects them via bitfield ICLSEL in its channel control register GxCHCTRx.

The global input class registers define the sample time and data conversion mode for each channel of any group that selects them via bitfield ICLSEL in its channel control register GxCHCTRx.

GxICLASS0 (x = 0 - 3)
Input Class Register 0, Group x

($x * 0400_H + 04A0_H$) **Reset Value: 0000 0000$_H$**

GxICLASS1 (x = 0 - 3)
Input Class Register 1, Group x

($x * 0400_H + 04A4_H$) **Reset Value: 0000 0000$_H$**

GLOBICLASSy (y = 0 - 1)
Input Class Register y, Global

($00A0_H + y * 0004_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**		**CME**		**0**	**0**	**0**			**STCE**		
r	r	r	r	r		rw		r	r	r			rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**		**CMS**		**0**	**0**	**0**			**STCS**		
r	r	r	r	r		rw		r	r	r			rw		

Field	Bits	Type	Description
STCS	[4:0]	rw	**Sample Time Control for Standard Conversions** Number of additional clock cycles to be added to the minimum sample phase of 2 analog clock cycles: Coding and resulting sample time see **Table 27-4**. For conversions of external channels, the value from bitfield STCE can be used.
0	[7:5]	r	**Reserved, write 0, read as 0**
CMS	[10:8]	rw	**Conversion Mode for Standard Conversions** 000_B 12-bit conversion 001_B 10-bit conversion 010_B 8-bit conversion 011_B Reserved 100_B Reserved 101_B 10-bit fast compare mode 110_B Reserved 111_B Reserved
0	[15:11]	r	**Reserved, write 0, read as 0**

Field	Bits	Type	Description
STCE	[20:16]	rw	**Sample Time Control for EMUX Conversions** Number of additional clock cycles to be added to the minimum sample phase of 2 analog clock cycles: Coding and resulting sample time see **Table 27-4**. For conversions of standard channels, the value from bitfield STCS is used.
0	[23:21]	r	**Reserved, write 0, read as 0**
CME	[26:24]	rw	**Conversion Mode for EMUX Conversions** 000_B 12-bit conversion 001_B 10-bit conversion 010_B 8-bit conversion 011_B Reserved 100_B Reserved 101_B 10-bit fast compare mode 110_B Reserved 111_B Reserved
0	[31:27]	r	**Reserved, write 0, read as 0**

Table 27-4 Sample Time Coding

STCS / STCE	Additional Clock Cycles	Sample Time
$0\ 0000_B$	0	2 / f_{ADCI}
$0\ 0001_B$	1	3 / f_{ADCI}
…	…	…
$0\ 1111_B$	15	17 / f_{ADCI}
$1\ 0000_B$	16	18 / f_{ADCI}
$1\ 0001_B$	32	34 / f_{ADCI}
…	…	…
$1\ 1110_B$	240	242 / f_{ADCI}
$1\ 1111_B$	256	258 / f_{ADCI}

27.7.2 Alias Feature

The Alias Feature redirects conversion requests for channels CH0 and/or CH1 to other channel numbers. This feature can be used to trigger conversions of the same input channel by independent events and to store the conversion results in different result registers.

- The same signal can be measured twice without the need to read out the conversion result to avoid data loss. This allows triggering both conversions quickly one after the other and being independent from CPU/DMA service request latency.
- The sensor signal is connected to only one analog input (instead of two analog inputs). This saves input pins in low-cost applications and only the leakage of one input has to be considered in the error calculation.
- Even if the analog input CH0 is used as alternative reference (see **Figure 27-13**), the internal trigger and data handling features for channel CH0 can be used.
- The channel settings for both conversions can be different (boundary values, service requests, etc.).

In typical low-cost AC-drive applications, only one common current sensor is used to determine the phase currents. Depending on the applied PWM pattern, the measured value has different meanings and the sample points have to be precisely located in the PWM period. **Figure 27-13** shows an example where the sensor signal is connected to one input channel (CHx) but two conversions are triggered for two different channels (CHx and CH0). With the alias feature, a conversion request for CH0 leads to a conversion of the analog input CHx instead of CH0, but taking into account the settings for CH0. Although the same analog input (CHx) has been measured, the conversion results can be stored and read out from the result registers RESx (conversion triggered for CHx) and RESy (conversion triggered for CH0). Additionally, different interrupts or limit boundaries can be selected, enabled or disabled.

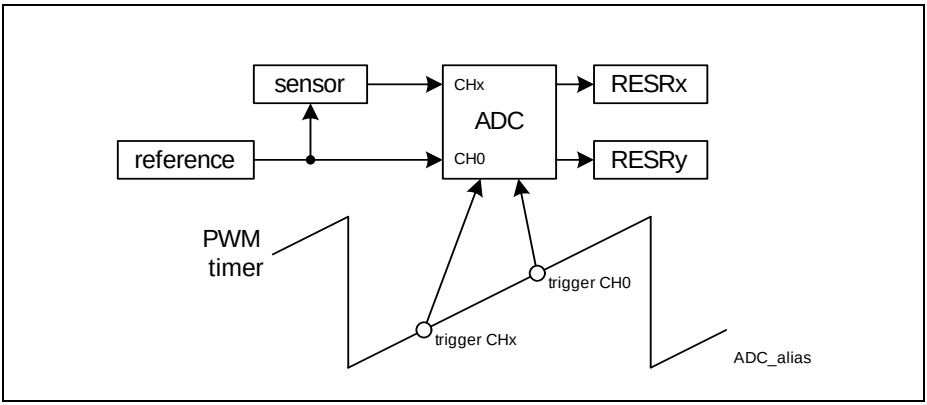
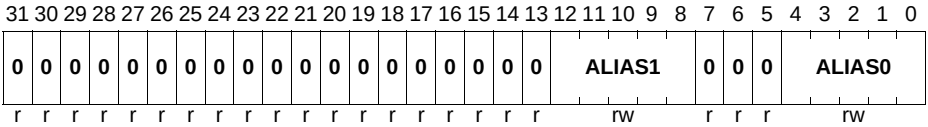


Figure 27-13 Alias Feature

Note: Use the alias feature to select the additional channels (CH12, CH13) for internal test signals. These channels cannot be selected directly.

The alias register can replace the channel numbers of channels CH0 and CH1 with another channel number. The reset value disables this redirection.

GxALIAS (x = 0 - 3)
Alias Register, Group x **($x * 0400_H + 04B0_H$)** **Reset Value: $0000\ 0100_H$**



Field	Bits	Type	Description
ALIAS0	[4:0]	rw	**Alias Value for CH0 Conversion Requests** Indicates the channel that is converted instead of channel CH0. The conversion is done with the settings defined for channel CH0.
0	[7:5]	r	**Reserved, write 0, read as 0**
ALIAS1	[12:8]	rw	**Alias Value for CH1 Conversion Requests** Indicates the channel that is converted instead of channel CH1. The conversion is done with the settings defined for channel CH1.
0	[31:13]	r	**Reserved, write 0, read as 0**

27.7.3 Conversion Modes

A conversion can be executed in several ways. The conversion mode is selected according to the requested resolution of the digital result and according to the acceptable conversion time (**Section 27.8**).
Use bitfield CMS/CME in register **GxICLASS0 (x = 0 - 3)** etc. to select a mode.

Standard Conversions

A standard conversion returns a result value with a predefined resolution. 8-bit, 10-bit, and 12-bit resolution can be selected.

These result values can be accumulated, filtered, or used for digital limit checking and determination of extrema.

Note: The calibrated converters can operate with and without post-calibration.

Fast Compare Mode

In Fast Compare Mode, the selected input voltage is directly compared with a digital value that is stored in the corresponding result register. This compare operation returns a binary result indicating if the compared input voltage is above or below the given reference value. This result is generated quickly and thus supports monitoring of boundary values.

Fast Compare Mode uses a 10-bit compare value stored left-aligned at bit position 11. Separate positive and negative delta values define an arbitrary hysteresis band.

Selecting Compare Values

Values for digital or analog compare operations can be selected from several sources. The separate GxBOUND registers provide software-defined compare values. Compare values can also be taken from a result register where it can be provided by another channel building a reference.

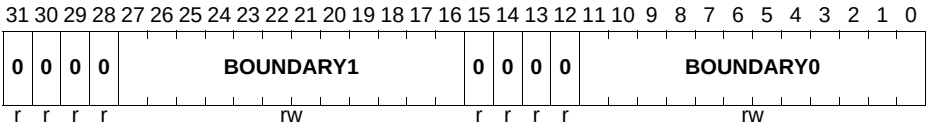
In Fast Compare Mode, the result registers provide the compare value while the bitfields of local GxBOUND registers define positive and negative delta values.

The local boundary register GxBOUND defines group-specific boundary values or delta limits for Fast Compare Mode.
The global boundary register GLOBBOUND defines general compare values for all channels.

Depending on the conversion width, the respective left 12/10/8 bits of a bitfield are used. For 10/8-bit results, the lower 2/4 bits must be zero!

GxBOUND (x = 0 - 3)
Boundary Select Register, Group x
($x * 0400_H + 04B8_H$) **Reset Value: 0000 0000$_H$**

GLOBBOUND
Global Boundary Select Register ($00B8_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
BOUNDARY0	[11:0]	rw	**Boundary Value 0 for Limit Checking** Standard Mode: This value is compared against the left-aligned conversion result. Fast Compare Mode: This value is added to the reference value (upper delta).
0	[15:12]	r	**Reserved, write 0, read as 0**
BOUNDARY1	[27:16]	rw	**Boundary Value 1 for Limit Checking** Standard Mode: This value is compared against the left-aligned conversion result. Fast Compare Mode: This value is subtracted from the reference value (lower delta).
0	[31:28]	r	**Reserved, write 0, read as 0**

27.7.4 Compare with Standard Conversions (Limit Checking)

The limit checking mechanism can automatically compare each digital conversion result to an upper and a lower boundary value. A channel event can then be generated when the result of a conversion/comparison is inside or outside a user-defined band (see bitfield CHEVMODE and **Figure 27-14**).

This feature supports automatic range monitoring and minimizes the CPU load by issuing service requests only under certain predefined conditions.

Note: Channel events can also be generated for each result value (ignoring the band) or they can be suppressed completely.

The boundary values to which results are compared can be selected from several sources (see register GxCHCTRy).

While bitfield BNDSELX = 0000_B, bitfields BNDSELU and BNDSELL select the valid upper/lower boundary value either from the group-specific boundary register **GxBOUND (x = 0 - 3)** or from the global boundary register **GLOBBOUND**. The group boundary register can be selected for each channel of the respective group, the global boundary register can be selected by each available channel.

Otherwise, the compare values are taken from result registers, where bitfield BNDSELX selects the upper boundary value (GxRES1 … GxRES15), the concatenated bitfields BNDSELU||BNDSELL select the lower boundary value (GxRES0 … GxRES15).

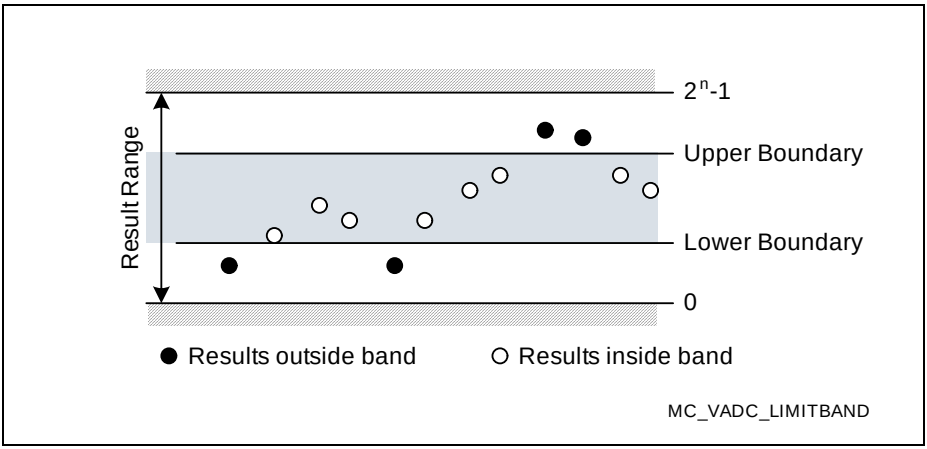


Figure 27-14 Result Monitoring through Limit Checking

A result value is considered inside the defined band when both of the following conditions are true:

- the value is less than or equal to the selected upper boundary
- the value is greater than or equal to the selected lower boundary

The result range can also be divided into two areas:
To select the lower part as valid band, set the lower boundary to the minimum value (000_H) and set the upper boundary to the highest intended value.
To select the upper part as valid band, set the upper boundary to the maximum value (FFF_H) and set the lower boundary to the lowest intended value.

Finding Extrema (Peak Detection)

The limit checking mechanism uses standard conversions and, therefore, always can provide the actual conversion result that was used for comparison. Combining this with a special FIFO mode, that only updates the corresponding FIFO stage if the result was above (or below) the current value of the stage, provides the usual conversion results and at the same time stores the highest (or lowest) result of a conversion sequence. For this operation the FIFO stage below the standard result must be selected as the compare value. Mode selection is done via bitfield FEN in register GxRCRy.

Before starting a peak detection sequence, write a reasonable start value to the result bitfield in the peak result register (e.g. 0000_H to find the maximum and $FFFF_H$ to find the minimum).

27.7.5 Utilizing Fast Compare Mode

In Fast Compare Mode, the input signal is directly compared to a value in bitfield RESULT in the associated result register. This comparison just provides a binary result (above/below), which is available in bit FCR in the same result register. If the exact result value is not required, this saves conversion time. A channel event can then be generated when the input signal becomes higher (or lower) than the compare value (see bitfield CHEVMODE and **Figure 27-15**).

The compare value in Fast Compare Mode is taken from the result register. Bitfields BOUNDARY1 and BOUNDARY0 in register **GxBOUND (x = 0 - 3)** define delta limits in this case. These deltas are added to (or subtracted from) the original compare value and allow defining an arbitrary hysteresis band.

The actual used compare value depends on the Fast Compare Result FCR (see registers **G0RESy (y = 0 - 15)**, etc.):

GxRESy.FCR = 0: reference value + upper delta
(GxRESy.RESULT + GxBOUND.BOUNDARY0)
GxRESy.FCR = 1: reference value - lower delta
(GxRESy.RESULT - GxBOUND.BOUNDARY1)

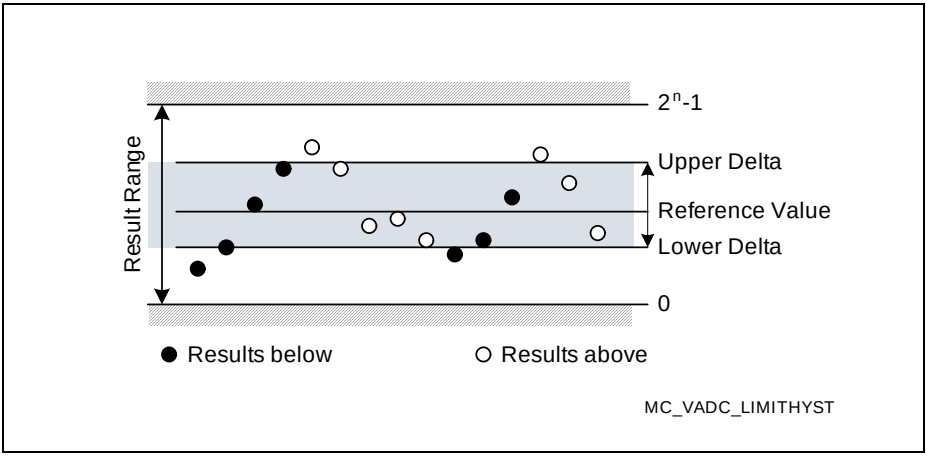


Figure 27-15 Result Monitoring through Compare with Hysteresis

27.7.6 Boundary Flag Control

Both limit checking mechanisms can be configured to automatically control the boundary flags. These boundary flags are also available as control signals for other modules. The flags can be set or cleared when the defined level is exceeded and the polarity of the output signal can be selected. A gate signal can be selected to enable the boundary flag operation while the gate is active.

Each boundary flag is available at group-specific output lines. Node pointers additionally route them to one of four boundary signals or one of the associated common service request lines, see register **GxBFLNP (x = 0 - 3)**.

For standard conversions, a boundary flag will be set/cleared when the conversion result is above the defined band, and will be cleared/set when the conversion result is below the defined band.

The band between the two boundary values defines a hysteresis for setting/clearing the boundary flags.

Using this feature on three channels that monitor linear hall elements can produce signals to feed the three hall position inputs of a unit that generates the corresponding PWM control signals.

In Fast Compare Mode, a boundary flag reflects the result of the comparisons, i.e. it will be set/cleared when the compared signal level is above the compare value, and will be cleared/set when the signal level is below the compare value. The delta values define a hysteresis band around the compare value.

Note: Clear register GxBOUND (i.e. the deltas) if a hysteresis is not wanted.

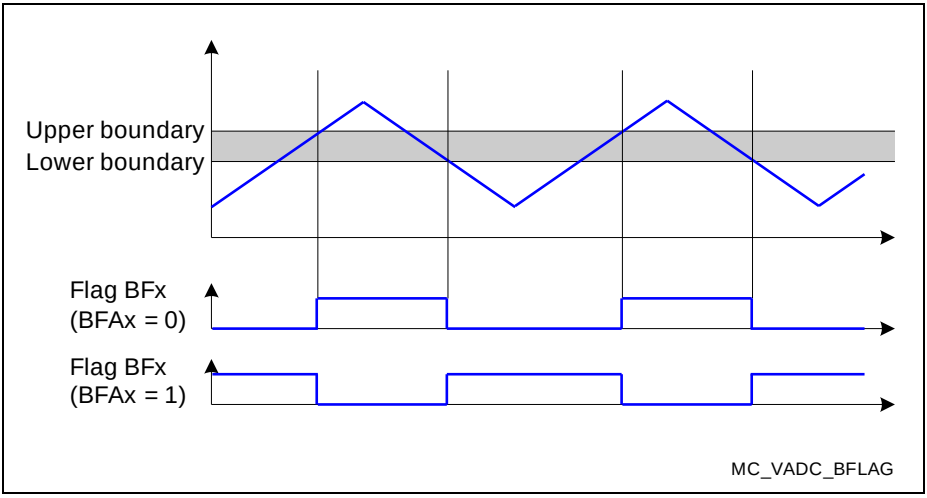


Figure 27-16 Boundary Flag Switching

Boundary flags can be switched by each compare operation, or the influence of compare operations can be restricted to the active phases of the corresponding request source gate signal (see **GxBFLC (x = 0 - 3)**).

Note: If a boundary flag is used together with Fast Compare Mode, it is recommended not to direct results from other channels to the corresponding result register.

The output signal derived from a boundary flag can be controlled by other modules. A select input and a data input are available to temporarily replace the boundary flag signal before sending it to the output pin (see **Figure 27-17**).

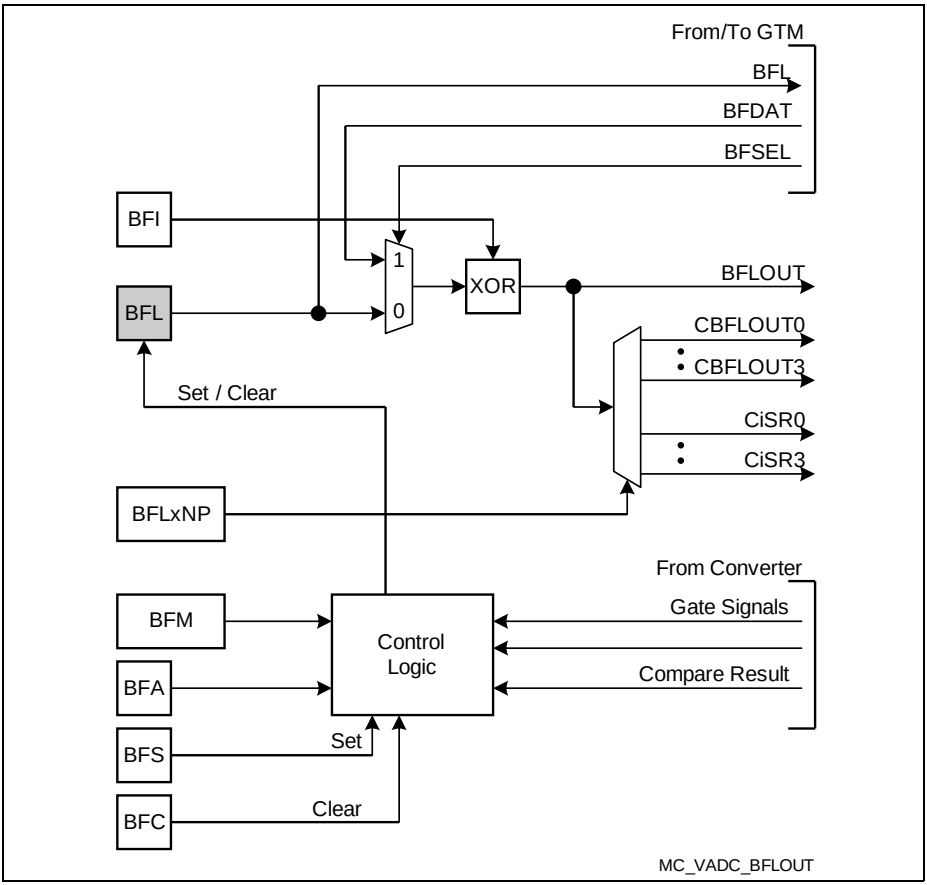


Figure 27-17 Boundary Flag Control

A boundary flag BFLy is assigned to result register GxRESy and thus to an arbitrary channel.

The Boundary Flag Register holds the boundary flags themselves together with bits to select the activation condition and the output signal polarity for each flag.

GxBFL (x = 0 - 3)
Boundary Flag Register, Group x

($x * 0400_H + 04C8_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**BFI 3**	**BFI 2**	**BFI 1**	**BFI 0**
r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**BFA 3**	**BFA 2**	**BFA 1**	**BFA 0**	**0**	**0**	**0**	**0**	**BFL 3**	**BFL 2**	**BFL 1**	**BFL 0**
r	r	r	r	rw	rw	rw	rw	r	r	r	r	rh	rh	rh	rh

Field	Bits	Type	Description
BFLy (y = 0 - 3)	y	rh	**Boundary Flag y** 0_B Passive state: result has not yet crossed the activation boundary (see bitfield BFAy), or selected gate signal is inactive, or this boundary flag is disabled 1_B Active state: result has crossed the activation boundary
0	[7:4]	r	**Reserved, write 0, read as 0**
BFAy (y = 0 - 3)	8 + y	rw	**Boundary Flag y Activation Select** 0_B Set boundary flag BFLy if result is above the defined band or compare value, clear if below 1_B Set boundary flag BFLy if result is below the defined band or compare value, clear if above
0	[15:12]	r	**Reserved, write 0, read as 0**
BFIy (y = 0 - 3)	16 + y	rw	**Boundary Flag y Inversion Control** 0_B Use BFLy directly 1_B Invert value and use $\overline{\text{BFLy}}$
0	[31:20]	r	**Reserved, write 0, read as 0**

The Boundary Flag Software Register provides means to set or clear each flag by software.

GxBFLS (x = 0 - 3)
Boundary Flag Software Register, Group x
($x * 0400_H + 04CC_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**BFS 3**	**BFS 2**	**BFS 1**	**BFS 0**
r	r	r	r	r	r	r	r	r	r	r	r	w	w	w	w

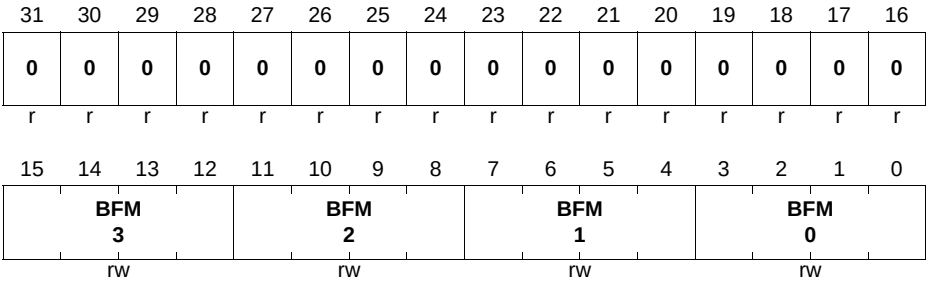
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**BFC 3**	**BFC 2**	**BFC 1**	**BFC 0**
r	r	r	r	r	r	r	r	r	r	r	r	w	w	w	w

Field	Bits	Type	Description
BFCy (y = 0 - 3)	y	w	**Boundary Flag y Clear** 0_B No action 1_B Clear bit BFLy
0	[15:4]	r	**Reserved, write 0, read as 0**
BFSy (y = 0 - 3)	16 + y	w	**Boundary Flag y Set** 0_B No action 1_B Set bit BFLy
0	[31:20]	r	**Reserved, write 0, read as 0**

Note: If a boundary flag is used together with Fast Compare Mode, it is recommended not to direct results from other channels to the corresponding result register.

The Boundary Flag Control Register selects the basic operation of the boundary flags.

GxBFLC (x = 0 - 3)
Boundary Flag Control Register, Group x
(x * 0400_H + $04D0_H$) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
BFM0, BFM1, BFM2, BFM3	[3:0], [7:4], [11:8], [15:12]	rw	**Boundary Flag y Mode Control** 0000_B Disable boundary flag, BFLy is not changed 0001_B Always enable boundary flag (follow compare results) 0010_B Enable boundary flag while gate of source 0 is active, clear BFLy while gate is inactive 0011_B Enable boundary flag while gate of source 1 is active, clear BFLy while gate is inactive 0100_B Enable boundary flag while gate of source 3 is active, clear BFLy while gate is inactive others: Reserved
0	[31:16]	r	**Reserved, write 0, read as 0**

The Boundary Flag Node Pointer Register directs signal GxBFLOUTy to alternate on-chip connections with other modules (in addition to the group-specific outputs). Possible targets are the corresponding common service request lines or the common boundary flag outputs (CBFLOUT0 … CBFLOUT3).

GxBFLNP (x = 0 - 3)
Boundary Flag Node Pointer Register, Group x
(x * 0400_H + $04D4_H$) **Reset Value: $0000\ FFFF_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BFL3NP				**BFL2NP**				**BFL1NP**				**BFL0NP**			
rw				rw				rw				rw			

Field	Bits	Type	Description
BFL0NP, BFL1NP, BFL2NP, BFL3NP	[3:0], [7:4], [11:8], [15:12]	rw	**Boundary Flag y Node Pointer** 0000_B Select common bondary flag output 0 … 0011_B Select common bondary flag output 3 0100_B Select shared service request line 0 … 0111_B Select shared service request line 3 1111_B Disabled, no common output signal others: Reserved *Note: For shared service request lines see common groups in **Table 27-8**.*
0	[31:16]	r	**Reserved, write 0, read as 0**

27.8 Conversion Timing

The total time required for a conversion comprises the time from the start of the sample phase[1] until the availability of the result.

The frequency at which conversions are triggered also depends on several configurable factors:

- The selected conversion time, according to the input class definitions. For conversions using an external multiplexer, also the extended sample times count.
- Delays induced by cancelled conversions that must be repeated.
- Delays due to equidistant sampling of other channels.
- The configured arbitration cycle time.
- The frequency of external trigger signals, if enabled.

The conversion timing depends on the following user-definable factors:

- The ADC conversion clock frequency, where $f_{ADCI} = f_{ADC} / (DIVA+1)$[2]
- The selected sample time, where $t_S = (2 + STC) \times t_{ADCI}$ (STC = additional sample time, see also **Table 27-4**)
- The selected operating mode (normal conversion / fast compare mode)
- The result width N (8/10/12 bits) for normal conversions
- The post-calibration time PC, if selected (PC = 2, otherwise 0)
- The selected duration of the MSB conversion (DM = 0 or 1)
- Synchronization steps done at module clock speed

The conversion time is the sum of sample time, conversion steps, and synchronization. It can be computed with the following formulas:

Standard conversions: $t_{CN} = (2 + STC + N + DM + PC) \times t_{ADCI} + 2 \times t_{ADC}$

Fast compare mode: $t_{CN} = (2 + STC + 2) \times t_{ADCI} + 2 \times t_{ADC}$

Timing Examples

System assumptions:
f_{ADC} = 100 MHz i.e. t_{ADC} = 10 ns, DIVA = 5, f_{ADCI} = 16.67 MHz i.e. t_{ADCI} = 60 ns
According to the given formula the following minimum conversion times can be achieved:

12-bit calibrated conversion:
$t_{CN12C} = (2 + 12 + 2) \times t_{ADCI} + 2 \times t_{ADC} = 16 \times 60\ ns + 2 \times 10\ ns = 980\ ns$

10-bit uncalibrated conversion:
$t_{CN10} = (2 + 10) \times t_{ADCI} + 2 \times t_{ADC} = 12 \times 60\ ns + 2 \times 10\ ns = 740\ ns$

Fast comparison:
$t_{FCM} = (2 + 2) \times t_{ADCI} + 2 \times t_{ADC} = 4 \times 60\ ns + 2 \times 10\ ns = 260\ ns$

1) The time from the trigger event that requests the corresponding conversion until the start of the sample phase depends on the arbitration and can, therefore, only be determined when the system setup is known.

2) The minimum prescaler factor for calibrated converters is 2.

27.9 Conversion Result Handling

The A/D converters can preprocess the conversions result data to a certain extent before storing them for retrieval by the CPU or a DMA channel. This supports the subsequent handling of result data by the application software.

Conversion result handling comprises the following functions:

- **Storage of Conversion Results** to user-configurable registers
- **Data Alignment** according to result width and endianess
- **Wait-for-Read Mode** to avoid loss of data
- **Result Event Generation**
- Data reduction or anti-aliasing filtering (see **Section 27.9.6**)

27.9.1 Storage of Conversion Results

The conversion result values of a certain group can be stored in one of the 16 associated group result registers or in the common global result register (can be used, for example, for the channels of the background source (see **Selecting a Result Register**).

This structure provides different locations for the conversion results of different sets of channels. Depending on the application needs (data reduction, auto-scan, alias feature, etc.), the user can distribute the conversion results to minimize CPU load and/or optimize the performance of DMA transfers.

Each result register has an individual data valid flag (VF) associated with it. This flag indicates when “new” valid data has been stored in the corresponding result register and can be read out.

For standard conversions, result values are available in bitfield RESULT. Conversions in Fast Compare Mode use bitfield RESULT for the reference value, so the result of the operation is stored in bit FCR.

Result registers can be read via two different views. These views use different addresses but access the same register data:

- When a result register is read via the **application view**, the corresponding valid flag is automatically cleared when the result is read. This provides an easy handshake between result generation and retrieval. This also supports wait-for-read mode.
- When a result register is read via the **debug view**, the corresponding valid flag remains unchanged when the result is read. This supports debugging by delivering the result value without disturbing the handshake with the application.

The application can retrieve conversion results through several result registers:

- Group result register:
  Returns the result value and the channel number
- Global result register:
  Returns the result value and the channel number and the group number

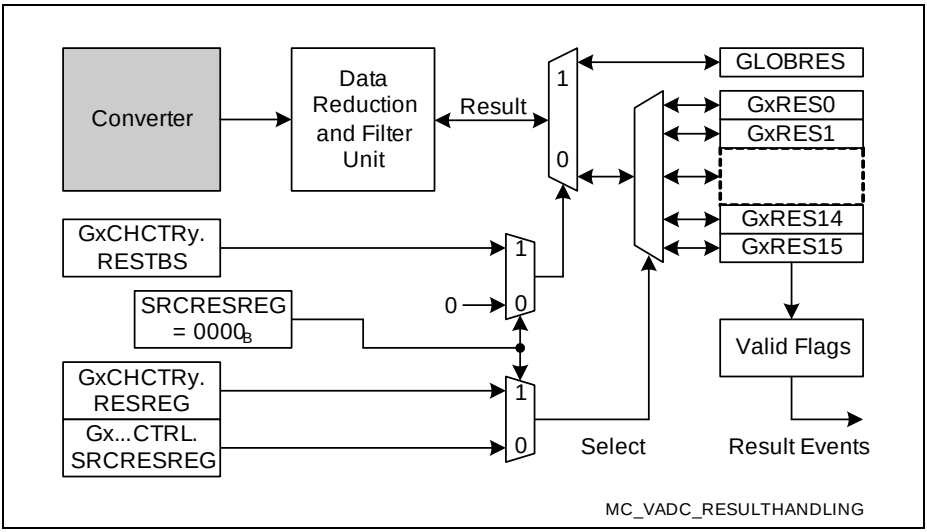


Figure 27-18 Conversion Result Storage

Selecting a Result Register

Conversion results are stored in result registers that can be assigned by the user according to the requirements of the application. The following bitfields direct the results to a register:

- SRCRESREG in register **GxQCTRL0 (x = 0 - 3)**, **GxASCTRL (x = 0 - 3)** or **BRSCTRL**
  Selects the group-specific result register GxRES1 … GxRES15
  when source-specific result registers are used
- RESTBS in register GxCHCTRy
  Selects the global result register for results requested by the background source
- RESREG in register GxCHCTRy
  Selects the group-specific result register GxRES0 … GxRES15
  when channel-specific result registers are used (see below).

Using source-specific result registers allows separating results from the same channel that are requested by different request sources. Usually these request sources are used by different tasks and are triggered at different times.

The group result control registers select the behavior of the result registers of a given group.

G0RCRy (y = 0 - 15)
Group 0 Result Control Reg. y (0680$_H$ + y * 0004$_H$) Reset Value: 0000 0000$_H$
G1RCRy (y = 0 - 15)
Group 1 Result Control Reg. y (0A80$_H$ + y * 0004$_H$) Reset Value: 0000 0000$_H$
G2RCRy (y = 0 - 15)
Group 2 Result Control Reg. y (0E80$_H$ + y * 0004$_H$) Reset Value: 0000 0000$_H$
G3RCRy (y = 0 - 15)
Group 3 Result Control Reg. y (1280$_H$ + y * 0004$_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRG EN	0	0	0	0	FEN		WFR	0	0	DMM		DRCTR			
rw	r	r	r	r	rw		rw	r	r	rw		rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
0	[15:0]	r	**Reserved, write 0, read as 0**
DRCTR	[19:16]	rw	**Data Reduction Control** Defines how result values are stored/accumulated in this register for the final result. The data reduction counter DRC can be loaded from this bitfield. The function of bitfield DRCTR is determined by bitfield DMM.
DMM	[21:20]	rw	**Data Modification Mode** 00_B Standard data reduction (accumulation) 01_B Result filtering mode[1)] 10_B Difference mode 11_B Reserved See **"Data Modification" on Page 27-125**
0	[23:22]	r	**Reserved, write 0, read as 0**
WFR	24	rw	**Wait-for-Read Mode Enable** 0_B Overwrite mode 1_B Wait-for-read mode enabled for this register

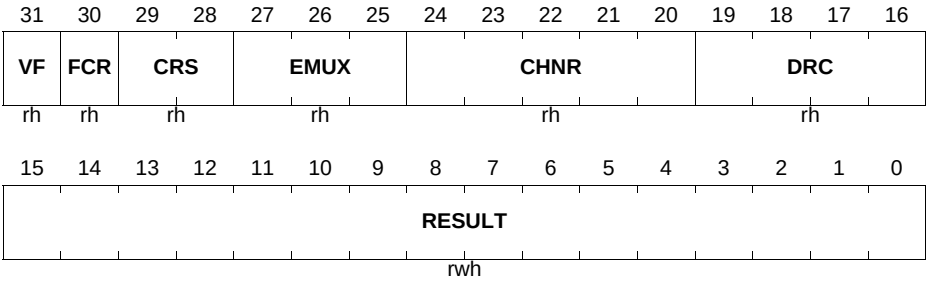
Field	Bits	Type	Description
FEN	[26:25]	rw	**FIFO Mode Enable** 00_B Separate result register 01_B Part of a FIFO structure: copy each new valid result 10_B Maximum mode: copy new result if bigger 11_B Minimum mode: copy new result if smaller
0	[30:27]	r	**Reserved, write 0, read as 0**
SRGEN	31	rw	**Service Request Generation Enable** 0_B No service request 1_B Service request after a result event

1) The filter registers are cleared while bitfield DMM ≠ 01_B.

The group result registers provide a selectable storage location for all channels of a given group.

Note: The preset value used in fast compare mode is written to the respective result register. The debug result registers are not writable.

G0RESy (y = 0 - 15)
Group 0 Result Register y **(0700_H + y * 0004_H)** **Reset Value: $0000\ 0000_H$**
G1RESy (y = 0 - 15)
Group 1 Result Register y **($0B00_H$ + y * 0004_H)** **Reset Value: $0000\ 0000_H$**
G2RESy (y = 0 - 15)
Group 2 Result Register y **($0F00_H$ + y * 0004_H)** **Reset Value: $0000\ 0000_H$**
G3RESy (y = 0 - 15)
Group 3 Result Register y **(1300_H + y * 0004_H)** **Reset Value: $0000\ 0000_H$**



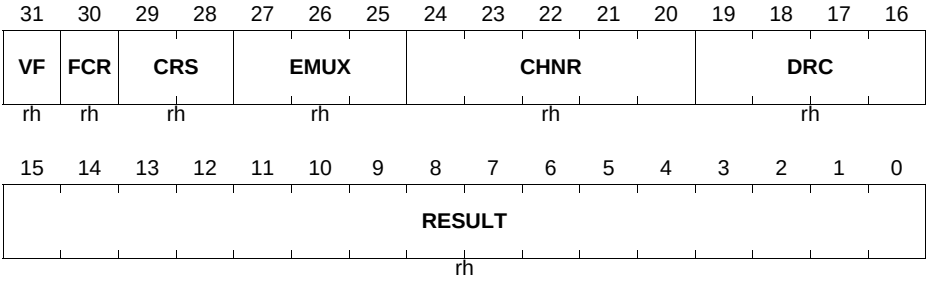
Field	Bits	Type	Description
RESULT	[15:0]	rwh	**Result of Most Recent Conversion** The position of the result bits within this bitfield depends on the configured operating mode. Please, refer to **Section 27.9.2**.
DRC	[19:16]	rh	**Data Reduction Counter** Indicates the number of values still to be accumulated for the final result. The final result is available and valid flag VF is set when bitfield DRC becomes zero (by decrementing or by reload). See **"Data Modification" on Page 27-125**
CHNR	[24:20]	rh	**Channel Number** Indicates the channel number corresponding to the value in bitfield RESULT.

Field	Bits	Type	Description
EMUX	[27:25]	rh	**External Multiplexer Setting** Indicates the setting of the external multiplexer, corresponding to the value in bitfield RESULT. *Note: Available in GxRES0 only. Use GxRES0 if EMUX information is required.*
CRS	[29:28]	rh	**Converted Request Source** Indicates the request source that as requested the conversion to which the result value in bitfield RESULT belongs. 00_B Request source 0 01_B Request source 1 10_B Request source 2 11_B Request source 3 or synchronized conversion[1)]
FCR	30	rh	**Fast Compare Result** Indicates the result of an operation in Fast Compare Mode. 0_B Signal level was below compare value 1_B Signal level was above compare value
VF	31	rh	**Valid Flag** Indicates a new result in bitfield RESULT or bit FCR. 0_B No new result available 1_B Bitfield RESULT has been updated with new result value and has not yet been read, or bit FCR has been updated

1) If request source 3 is available/active and/or synchronized conversions are configured.

The debug view of the group result registers provides access to all result registers of a given group, however, without clearing the valid flag.

G0RESDy (y = 0 - 15)
Group 0 Result Reg. y, Debug (0780_H + y * 0004_H) Reset Value: $0000\ 0000_H$
G1RESDy (y = 0 - 15)
Group 1 Result Reg. y, Debug ($0B80_H$ + y * 0004_H) Reset Value: $0000\ 0000_H$
G2RESDy (y = 0 - 15)
Group 2 Result Reg. y, Debug ($0F80_H$ + y * 0004_H) Reset Value: $0000\ 0000_H$
G3RESDy (y = 0 - 15)
Group 3 Result Reg. y, Debug (1380_H + y * 0004_H) Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
RESULT	[15:0]	rh	**Result of Most Recent Conversion** The position of the result bits within this bitfield depends on the configured operating mode. Please, refer to **Section 27.9.2**.
DRC	[19:16]	rh	**Data Reduction Counter** Indicates the number of values still to be accumulated for the final result. The final result is available and valid flag VF is set when bitfield DRC becomes zero (by decrementing or by reload). See **"Data Modification" on Page 27-125**
CHNR	[24:20]	rh	**Channel Number** Indicates the channel number corresponding to the value in bitfield RESULT.
EMUX	[27:25]	rh	**External Multiplexer Setting** Indicates the setting of the external multiplexer, corresponding to the value in bitfield RESULT. *Note: Available in GxRESD0 only. Use GxRESD0 if EMUX information is required.*

Field	Bits	Type	Description
CRS	[29:28]	rh	**Converted Request Source** Indicates the request source that as requested the conversion to which the result value in bitfield RESULT belongs. 00_B Request source 0 01_B Request source 1 10_B Request source 2 11_B Request source 3 or synchronized conversion[1)]
FCR	30	rh	**Fast Compare Result** Indicates the result of an operation in Fast Compare Mode. 0_B Signal level was below compare value 1_B Signal level was above compare value
VF	31	rh	**Valid Flag** Indicates a new result in bitfield RESULT or bit FCR. 0_B No new result available 1_B Bitfield RESULT has been updated with new result value and has not yet been read, or bit FCR has been updated

1) If request source 3 is available/active and/or synchronized conversions are configured.

The global result control register selects the behavior of the global result register.

GLOBRCR
Global Result Control Register (0280_H) Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRGEN	**0**	**0**	**0**	**0**	**0**	**0**	**WFR**	**0**	**0**	**0**	**0**	**DRCTR**			
rw	r	r	r	r	r	r	rw	r	r	r	r	rw			

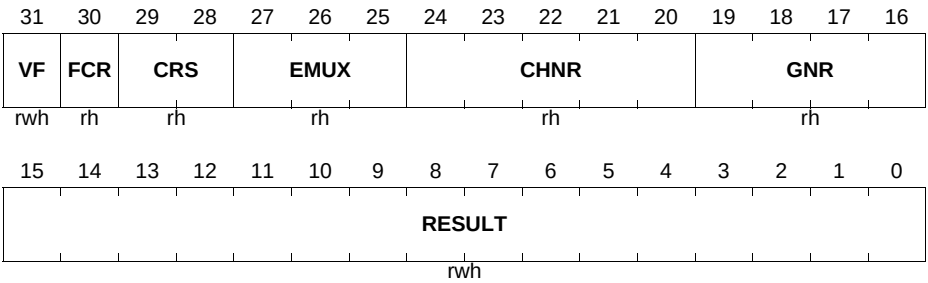
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
0	[15:0]	r	**Reserved, write 0, read as 0**
DRCTR	[19:16]	rw	**Data Reduction Control** Defines how result values are stored/accumulated in this register for the final result. The data reduction counter DRC can be loaded from this bitfield. 0000_BData reduction disabled others: see **"Function of Bitfield DRCTR" on Page 27-125**[1)]
0	[23:20]	r	**Reserved, write 0, read as 0**
WFR	24	rw	**Wait-for-Read Mode Enable** 0_B Overwrite mode 1_B Wait-for-read mode enabled for this register
0	[30:25]	r	**Reserved, write 0, read as 0**
SRGEN	31	rw	**Service Request Generation Enable** 0_B No service request 1_B Service request after a result event

1) Only standard data reduction is available for the global result register, i.e. DMM is assumed as 00_B.

The global result register provides a common storage location for all channels of all groups.

GLOBRES
Global Result Register **(0300_H)** **Reset Value: 0000 0000_H**
GLOBRESD
Global Result Register, Debug **(0380_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
RESULT	[15:0]	rwh	**Result of most recent conversion** The position of the result bits within this bitfield depends on the configured operating mode.[1)] Please, refer to **Section 27.9.2**.
GNR	[19:16]	rh	**Group Number** Indicates the group to which the channel number in bitfield CHNR refers.
CHNR	[24:20]	rh	**Channel Number** Indicates the channel number corresponding to the value in bitfield RESULT.
EMUX	[27:25]	rh	**External Multiplexer Setting** Indicates the setting of the external multiplexer, corresponding to the value in bitfield RESULT.
CRS	[29:28]	rh	**Converted Request Source** Indicates the request source that as requested the conversion to which the result value in bitfield RESULT belongs.
FCR	30	rh	**Fast Compare Result** Indicates the result of an operation in Fast Compare Mode. 0_B Signal level was below compare value 1_B Signal level was above compare value
VF	31	rwh	**Valid Flag** Indicates a new result in bitfield RESULT or bit FCR. 0_B Read access: No new valid data available Write access: No effect 1_B Read access: Bitfield RESULT contains valid data and has not yet been read, or bit FCR has been updated Write access: Clear this valid flag and the data reduction counter (overrides a hardware set action)[1)]

1) Only writable in register GLOBRES, not in register GLOBRESD.

The valid flag register summarizes the valid flags of all result registers.

GxVFR (x = 0 - 3)
Valid Flag Register, Group x **(x * 0400_H + $05F8_H$)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VF15	**VF14**	**VF13**	**VF12**	**VF11**	**VF10**	**VF9**	**VF8**	**VF7**	**VF6**	**VF5**	**VF4**	**VF3**	**VF2**	**VF1**	**VF0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
VFy **(y = 0 - 15)**	y	rwh	**Valid Flag of Result Register x** Indicates a new result in bitfield RESULT or in bit FCR. 0_B Read access: No new valid data available Write access: No effect 1_B Read access: Result register x contains valid data and has not yet been read, or bit FCR has been updated Write access: Clear this valid flag and bitfield DRC in register GxRESy (overrides a hardware set action)
0	[31:16]	r	**Reserved, write 0, read as 0**

27.9.2 Data Alignment

The position of a conversion result value within the selected result register depends on 3 configurations (summary in **Figure 27-19**):

- The selected result width (12/10/8 bits, selected by the conversion mode)
- The selected result position (Left/Right-aligned
- The selected data accumulation mode (data reduction)

These options provide the conversion results in a way that minimizes data handling for the application software.

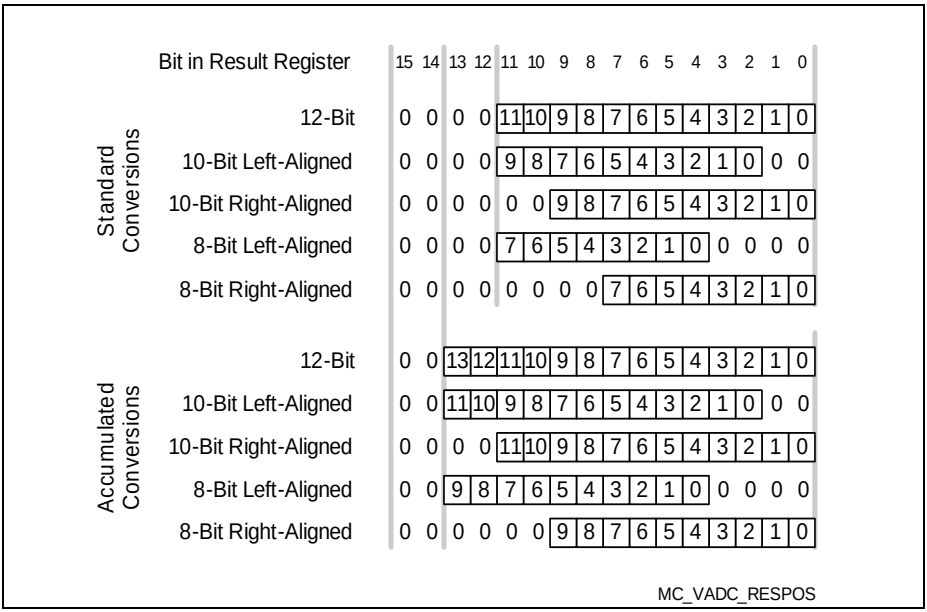


Figure 27-19 Result Storage Options

Bitfield RESULT can be written by software to provide the reference value for Fast Compare Mode. In this mode, bits 11-2 are evaluated, the other bits are ignored.

27.9.3 Wait-for-Read Mode

The wait-for-read mode prevents data loss due to overwriting a result register with a new conversion result before the CPU (or DMA) has read the previous data. For example, auto-scan conversion sequences or other sequences with “relaxed” timing requirements may use a common result register. However, the results come from different input channels, so an overwrite would destroy the result from the previous conversion[1)].

Wait-for-read mode automatically suspends the start of a conversion for this channel from this source until the current result has been read. So a conversion or a conversion sequence can be requested by a hardware or software trigger, while each conversion is only started after the result of the previous one has been read. This automatically aligns the conversion sequence with the CPU/DMA capability to read the formerly converted result (latency).

If wait-for-read mode is enabled for a result register (bit GxRCRy.WFR = 1), a request source does not generate a conversion request while the targeted result register contains valid data (indicated by the valid flag VF = 1) or if a currently running conversion targets the same result register.

If two request sources target the same result register with wait-for-read mode selected, a higher priority source cannot interrupt a lower priority conversion request started before the higher priority source has requested its conversion. Cancel-inject-repeat mode does not work in this case. In particular, this must be regarded if one of the involved sources is the background source (which usually has lowest priority). If the higher priority request targets a different result register, the lower priority conversion can be cancelled and repeated afterwards.

*Note: Wait-for-read mode is ignored for synchronized conversions of synchronization slaves (see **Section 27.10**).*

1) Repeated conversions of a single channel that use a separate result register will not destroy other results, but rather update their own previous result value. This way, always the actual signal data is available in the result register.

27.9.4 Result FIFO Buffer

Result registers can either be used as direct target for conversion results or they can be concatenated with other result registers of the same ADC group to form a result FIFO buffer (first-in-first-out buffer mechanism). A result FIFO stores several measurement results that can be read out later with a “relaxed” CPU response timing. It is possible to set up more than one FIFO buffer structure with the available result registers.

Result FIFO structures of two or more registers are built by concatenating result registers to their following “neighbor” result register (with next higher index, see **Figure 27-20**). This is enabled by setting bitfield GxRCRy.FEN = 01_B.
Conversion results are stored to the register with the highest index of a FIFO structure. Software reads the values from the FIFO register with the lowest index.

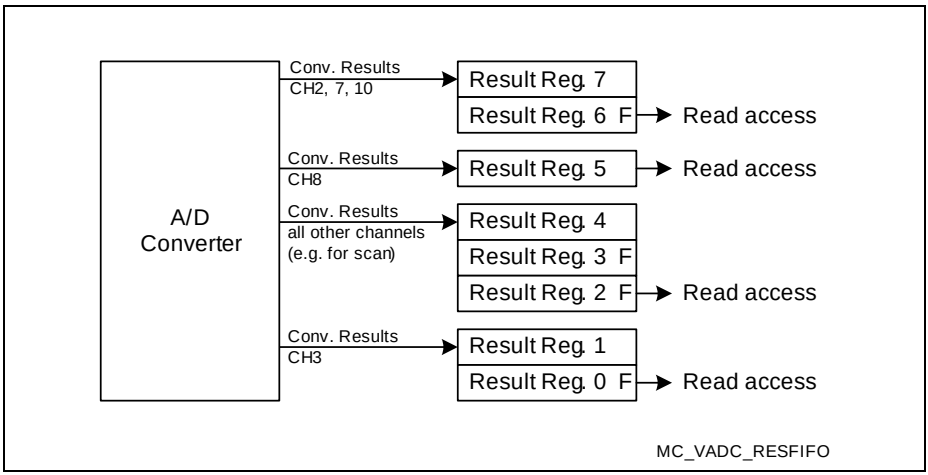


Figure 27-20 Result FIFO Buffers

In the example shown the result registers have been configured in the following way:

- 2-stage buffer consisting of result registers 7-6
- dedicated result register 5
- 3-stage buffer consisting of result registers 4-3-2
- 2-stage buffer consisting of result registers 1-0

Table 27-5 summarizes the required configuration of result registers if they are combined to build result FIFO buffers.

Table 27-5 Properties of Result FIFO Registers

Function	Input Stage	Intermed. Stage	Output Stage
Result target	YES	no	no
Application read	no	no	YES
Data reduction mode	YES	no	no
Wait-for-read mode	YES	no	no
Result event interrupt	no	no	YES
FIFO enable (FEN)	00_B	01_B	01_B
Registers in example	7, 4, 1	3	6, 2, 0

Note: If enabled, a result interrupt is generated for each data word in the FIFO.

27.9.5 Result Event Generation

A result event can be generated when a new value is stored to a result register. Result events can be restricted due to data accumulation and be generated only if the accumulation is complete.

Result events can also be suppressed completely.

27.9.6 Data Modification

The data resulting from conversions can be automatically modified before being used by an application. Several options can be selected (bitfield DMM in register **G0RCRy (y = 0 - 15)** etc.) which reduce the CPU/DMA load required to unload and/or process the conversion data.

- **Standard Data Reduction Mode (for GxRES0 … GxRES15):**
  Accumulates 2, 3, or 4 result values within each result register before generating a result interrupt. This can remove some noise from the input signal.
- **Result Filtering Mode (FIR, for GxRES7, GxRES15):**
  Applies a 3rd order Finite Impulse Response Filter (FIR) with selectable coefficients to the conversion results for the selected result register.
- **Result Filtering Mode (IIR, for GxRES7, GxRES15):**
  Applies a 1st order Infinite Impulse Response Filter (IIR) with selectable coefficients to the conversion results for the selected result register.
- **Difference Mode (for GxRES1 … GxRES15):**
  Subtracts the contents of result register GxRES0 from the conversion results for the selected result register. Bitfield DRCTR is not used in this mode.

Table 27-6 Function of Bitfield DRCTR

DRCTR	Standard Data Reduction Mode (DMM = 00_B)	DRCTR	Result Filtering Mode (DMM = 01_B)[1)]
0000_B	Data Reduction disabled	0000_B	FIR filter: a=2, b=1, c=0
0001_B	Accumulate 2 result values	0001_B	FIR filter: a=1, b=2, c=0
0010_B	Accumulate 3 result values	0010_B	FIR filter: a=2, b=0, c=1
0011_B	Accumulate 4 result values	0011_B	FIR filter: a=1, b=1, c=1
0100_B	Reserved	0100_B	FIR filter: a=1, b=0, c=2
0101_B	Reserved	0101_B	FIR filter: a=3, b=1, c=0
0110_B	Reserved	0110_B	FIR filter: a=2, b=2, c=0
0111_B	Reserved	0111_B	FIR filter: a=1, b=3, c=0
1000_B	Reserved	1000_B	FIR filter: a=3, b=0, c=1
1001_B	Reserved	1001_B	FIR filter: a=2, b=1, c=1
1010_B	Reserved	1010_B	FIR filter: a=1, b=2, c=1
1011_B	Reserved	1011_B	FIR filter: a=2, b=0, c=2
1100_B	Reserved	1100_B	FIR filter: a=1, b=1, c=2
1101_B	Reserved	1101_B	FIR filter: a=1, b=0, c=3

Table 27-6 Function of Bitfield DRCTR (cont'd)

DRCTR	Standard Data Reduction Mode (DMM = 00_B)	DRCTR	Result Filtering Mode (DMM = 01_B)[1]
1110_B	Reserved	1110_B	IIR filter: a=2, b=2
1111_B	Reserved	1111_B	IIR filter: a=3, b=4

1) The filter registers are cleared while bitfield DMM ≠ 01_B.

Standard Data Reduction Mode

The data reduction mode can be used as digital filter for anti-aliasing or decimation purposes. It accumulates a maximum of 4 conversion values to generate a final result.

Each result register can be individually enabled for data reduction, controlled by bitfield DRCTR in registers **G0RCRy (y = 0 - 15)**ff and **GLOBRCR**. The data reduction counter DRC indicates the actual status of the accumulation.

Note: Conversions for other result registers can be inserted between conversions to be accumulated.

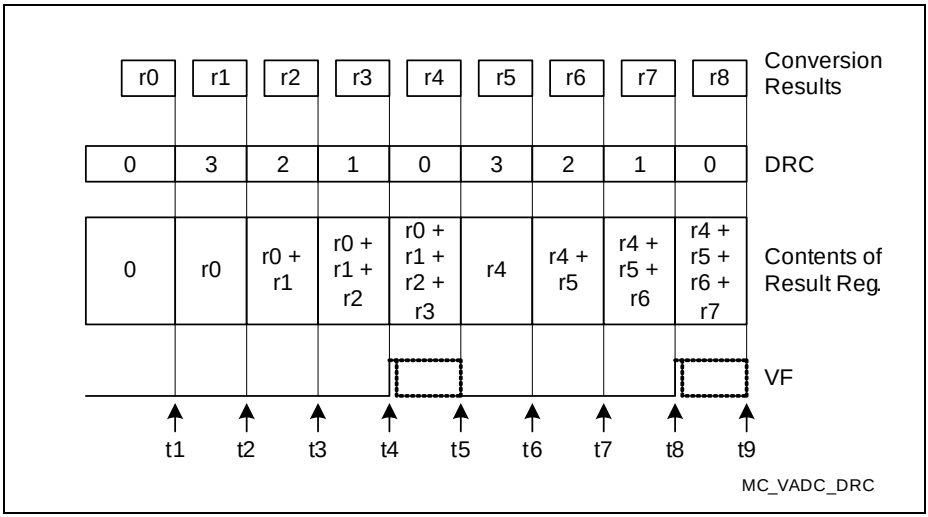


Figure 27-21 Standard Data Reduction Filter

This example shows a data reduction sequence of 4 accumulated conversion results. Eight conversion results (r0 … r7) are accumulated and produce 2 final results.

When a conversion is complete and stores data to a result register that has data reduction mode enabled, the data handling is controlled by the data reduction counter DRC:

- If DRC = 0 (t1, t5, t9 in the example), the conversion result is stored to the register. DRC is loaded with the contents of bitfield DRCTR (i.e. the accumulation begins).
- If DRC > 0 (t2, t3, t4 and t6, t7, t8 in the example), the conversion result is added to the value in the result register.
  DRC is decremented by 1.
- If DRC becomes 0, either decremented from 1 (t4 and t8 in the example) or loaded from DRCTR, the valid bit for the respective result register is set and a result register event occurs.

The final result must be read before the next data reduction sequence starts (before t5 or t9 in the example). This automatically clears the valid flag.

*Note: Software can clear the data reduction counter DRC by clearing the corresponding valid Flag (via **GxVFR (x = 0 - 3)**).*

The response time to read the final data reduction results can be increased by associating the adjacent result register to build a result FIFO (see **Figure 27-22**). In this case, the final result of a data reduction sequence is loaded to the adjacent register. The value can be read from this register until the next data reduction sequence is finished (t8 in the 2nd example).

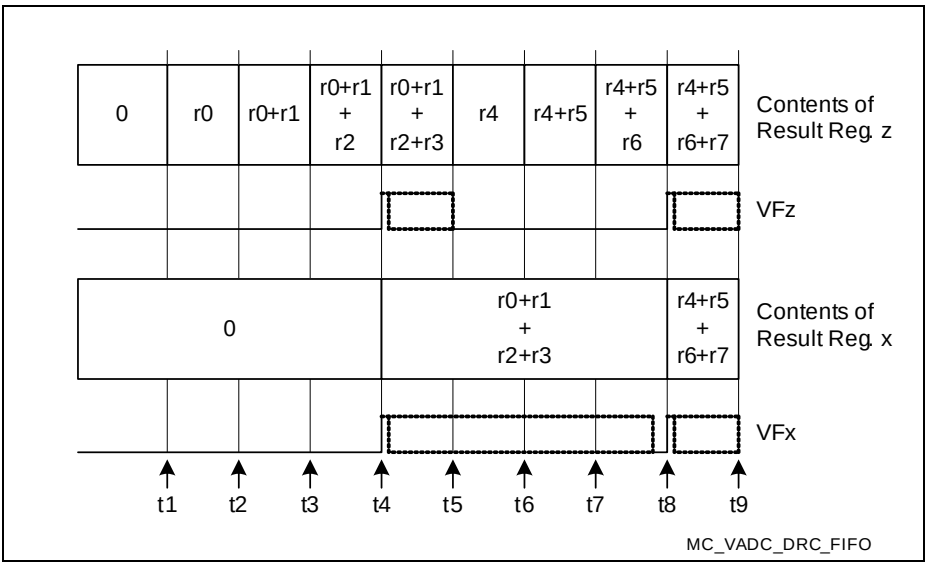


Figure 27-22 Standard Data Reduction Filter with FIFO Enabled

Finite Impulse Response Filter Mode (FIR)

The FIR filter (see **Figure 27-23**) provides 2 result buffers for intermediate results (RB1, RB2) and 3 configurable tap coefficients (a, b, c).

The conversion result and the intermediate result buffer values are added weighted with their respective coefficients to form the final value for the result register. Several predefined sets of coefficients can be selected via bitfield DRCTR (coding listed in **Table 27-6**) in registers **G0RCRy (y = 0 - 15)**ff and **GLOBRCR**. These coefficients lead to a gain of 3 or 4 to the ADC result producing a 14-bit value. The valid flag (VF) is activated for each sample after activation, i.e. for each sample generates a valid result.

Note: Conversions for other result registers can be inserted between conversions to be filtered.

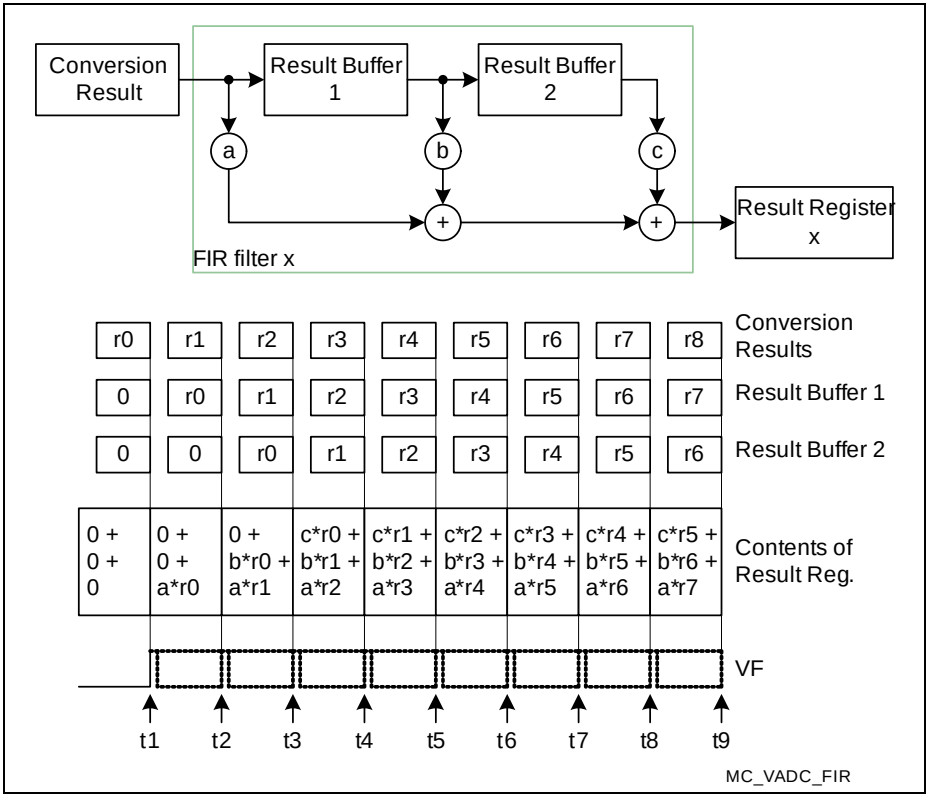


Figure 27-23 FIR Filter Structure

Note: The filter registers are cleared while bitfield DMM ≠ 01_B.

Infinite Impulse Response Filter Mode (IIR)

The IIR filter (see **Figure 27-24**) provides a result buffer (RB) and 2 configurable coefficients (a, b). It represents a first order low-pass filter.

The conversion result, weighted with the respective coefficient, and a fraction of the previous result are added to form the final value for the result register. Several predefined sets of coefficients can be selected via bitfield DRCTR (coding listed in **Table 27-6**) in registers **G0RCRy (y = 0 - 15)**ff and **GLOBRCR**. These coefficients lead to a gain of 4 to the ADC result producing a 14-bit value. The valid flag (VF) is activated for each sample after activation, i.e. for each sample generates a valid result.

Note: Conversions for other result registers can be inserted between conversions to be filtered.

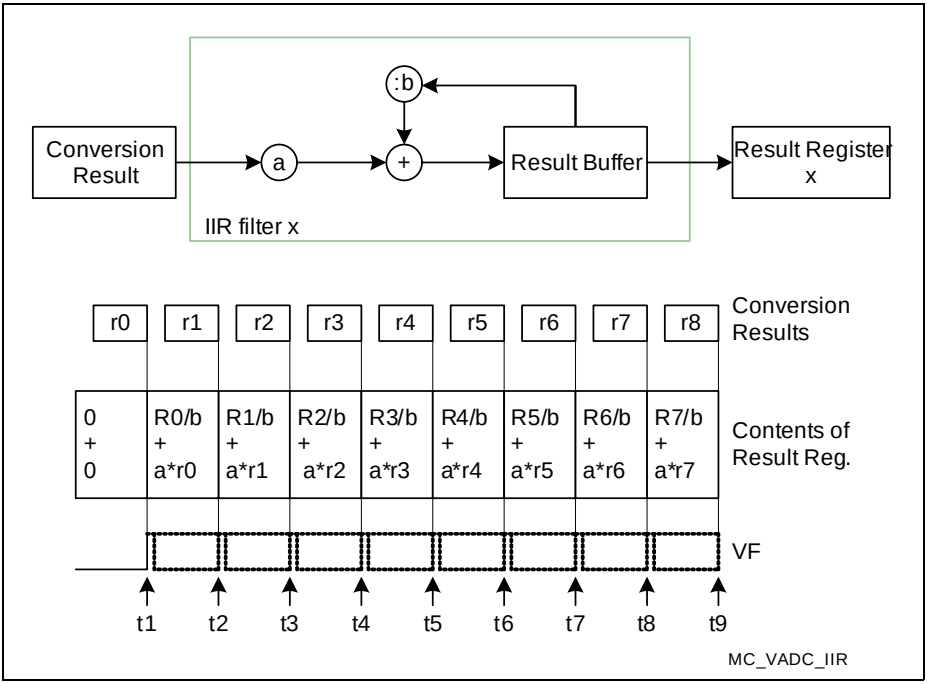


Figure 27-24 IIR Filter Structure

Note: The filter registers are cleared while bitfield DMM ≠ 01_B.

Difference Mode

Subtracting the contents of result register 0 from the actual result puts the results of the respective channel in relation to another signal. No software action is required.

The reference channel must store its result(s) into result register 0. The reference value can be determined once and then be used for a series of conversions, or it can be converted before each related conversion.

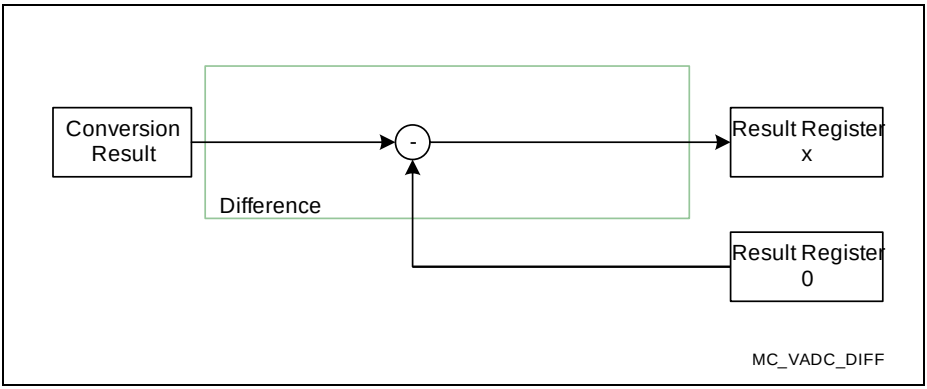


Figure 27-25 Result Difference

27.10 Synchronization of Conversions

The conversions of an ADC kernel can be scheduled either self-timed according to the kernel's configuration or triggered by external (outside the ADC) signals:
Synchronized conversions support parallel conversion of channels within a synchronization group[1]. This optimizes e.g. the control of electrical drives.
Equidistant sampling supports conversions in a fixed raster with minimum jitter. This optimizes e.g. filter algorithms or audio applications.

27.10.1 Synchronized Conversions for Parallel Sampling

Several independent ADC kernels[1] implemented in the TC21x/TC22x/TC23x can be synchronized for simultaneous measurements of analog input channels. While no parallel conversion is requested, the kernels can work independently.

The synchronization mechanism for parallel conversions ensures that the sample phases of the related channels start simultaneously. Synchronized kernels convert the same channel that is requested by the master. Different values for the resolution and the sample phase length of each kernel for a parallel conversion are supported.

A parallel conversion can be requested individually for each input channel (one or more). In the example shown in **Figure 27-26**, input channels CH3 of the ADC kernels 0 and 1 are converted synchronously, whereas other input channels do not lead to parallel conversions.

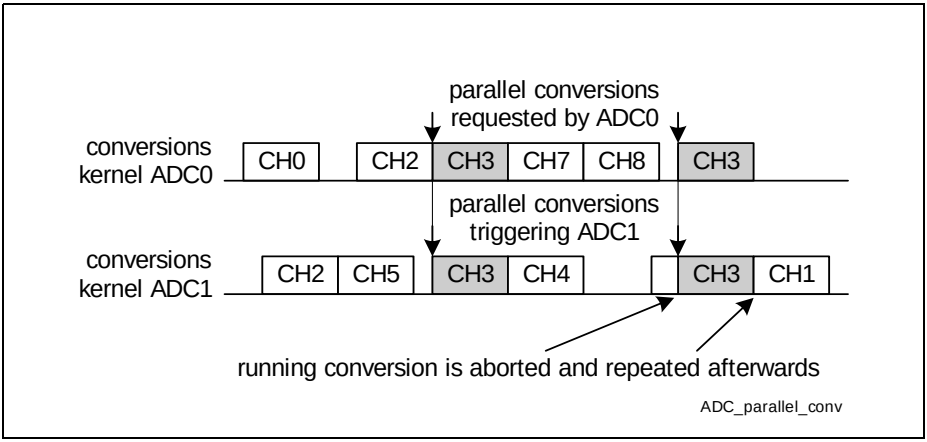


Figure 27-26 Parallel Conversions

1) For a summary, please refer to **"Synchronization Groups in the TC21x/TC22x/TC23x" on Page 27-168**.

One kernel operates as synchronization master, the other kernel(s) operate(s) as synchronization slave. Each kernel can play either role. The arbiters of all involved kernels must run synchronously. This is achieved by switching off all involved kernels before the initialization and switching on the master kernel at the end of the initialization sequence.

Master and slave kernels form a “conversion group” to control parallel sampling:

- The arbiters must run permanently (bits **GxARBCFG (x = 0 - 3)**.ARBM = 0). Initialize the slave before the master to have the arbiters run synchronously. Set the master’s GxARBCFG.ANONC at the end of the initialization.
- **The synchronization master** controls the slave(s) by providing the control information **GxARBCFG (x = 0 - 3)**.ANONS (see **Figure 27-27**) and the requested channel number.
  - Bitfield **GxSYNCTR (x = 0 - 3)**.STSEL = 00_B selects the master’s ANON information as the source of the ANON information for all kernels of the synchronization group.[1)]
  - The ready signals indicate when a slave kernel is ready to start the sample phase of a parallel conversion. Bit **GxSYNCTR (x = 0 - 3)**.EVALRy = 1 enables the control by the ready signal (in the example kernel 1 is the slave, so EVALR1 = 1).
  - The master requests a synchronized conversion of a certain channel (SYNC = 1 in the corresponding channel control register GxCHCTRy), which is also requested in the connected slave ADC kernel(s).
  - Wait-for-read mode is supported for the master.
- **The synchronization slave** reacts to incoming synchronized conversion requests from the master. While no synchronized conversions are requested, the slave kernel can execute “local” conversions.
  - Bitfield **GxSYNCTR (x = 0 - 3)**.STSEL = 01_B/10_B/11_B selects the master’s ANON information as the source of the ANON information for all kernels of the synchronization group[1)] (in the example kernel 0 is the master, so STSEL = 01_B).
  - The ready signals indicate when the master kernel and the other slave kernels are ready to start the sample phase of a parallel conversion. Bit **GxSYNCTR (x = 0 - 3)**.EVALRy = 1 enables the control by the ready signal (in the example kernel 0 is the master, so EVALR1 = 1).
  - The slave timing must be configured according to the master timing (ARBRND in register **GxARBCFG (x = 0 - 3)**) to enable parallel conversions.
  - A parallel conversion request is always handled with highest priority and cancel-inject-repeat mode.
  - Wait-for-read mode is ignored in the slave. Previous results may be overwritten, in particular, if the same result register is used by other conversions.

1) STSEL = 00_B selects the own ANON information. The other control inputs (STSEL = 01_B/10_B/11_B) are connected to the other kernels of a synchronization group in ascending order (see also **Table 27-9 “Synchronization Groups in the TC21x/TC22x/TC23x” on Page 27-168**).

- Once started, a parallel conversion cannot be aborted.

*Note: Synchronized conversions request the same channel number, defined by the master. Using the alias feature (see **Section 27.7.2**), analog signals from different input channels can be converted. This is advantageous if e.g. CH0 is used as alternate reference.*

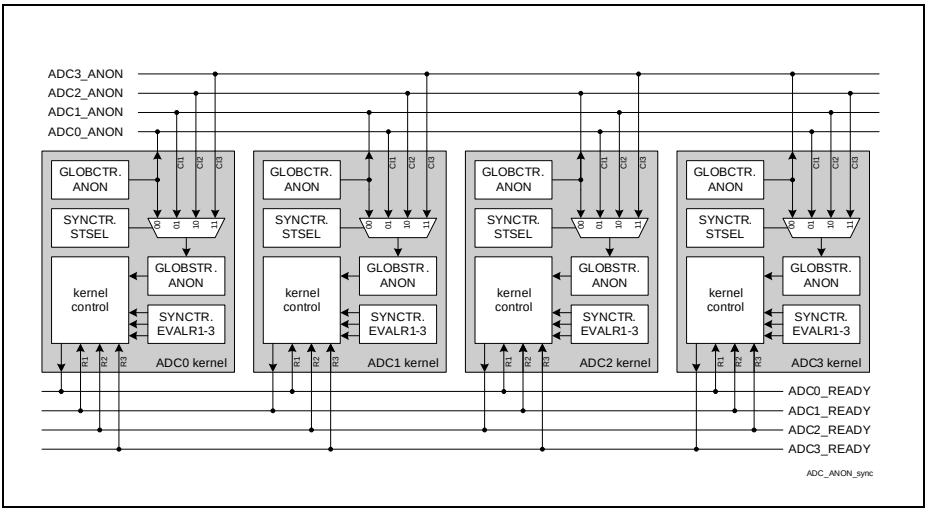


Figure 27-27 Synchronization via ANON and Ready Signals

The Synchronization Control Register controls the synchronization of kernels for parallel conversions.

Note: Program register GxSYNCTR only while bitfield GxARBCFG.ANONS = 00_B in all ADC kernels of the conversion group. Set the master's bitfield ANONC to 11_B afterwards.

GxSYNCTR (x = 0 - 3)
Synchronization Control Register, Group x
(x * 0400_H + $04C0_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	EVALR3	EVALR2	EVALR1	0	0	STSEL	
r	r	r	r	r	r	r	r	r	rw	rw	rw	r	r	rw	

Field	Bits	Type	Description
STSEL	[1:0]	rw	**Start Selection** Controls the synchronization mechanism of the ADC kernel. 00_B Kernel is synchronization master: Use own bitfield GxARBCFG.ANONC 01_B Kernel is synchronization slave: Control information from input CI1 10_B Kernel is synchronization slave: Control information from input CI2 11_B Kernel is synchronization slave: Control information from input CI3 *Note: Control inputs CIx see **Figure 27-27**, connected kernels see **Table 27-9**.*
0	[3:2]	r	**Reserved, write 0, read as 0**

Field	Bits	Type	Description
EVALR1, EVALR2, EVALR3	4, 5, 6	rw	**Evaluate Ready Input Rx** Enables the ready input signal for a kernel of a conversion group. 0_B No ready input control 1_B Ready input Rx is considered for the start of a parallel conversion of this conversion group
0	[31:7]	r	**Reserved, write 0, read as 0**

27.10.2 Equidistant Sampling

To optimize the input data e.g. for filter or audio applications, conversions can be executed in a fixed timing raster. Conversions for equidistant sampling are triggered by an external signal (e.g. a timer). To generate the trigger signal synchronous to the arbiter, the ADC provides an output signal (ARBCNT) that is activated once per arbitration round and serves as timing base for the trigger timer. In this case, the arbiter must run permanently (**GxARBCFG (x = 0 - 3)**.ARBM = 0). If the timer has an independent time base, the arbiter can be stopped while no requests are pending. The preface time (see **Figure 27-28**) must be longer than one arbitration round and the highest possible conversion time.

Select timer mode (TMEN = 1 in register **GxQCTRL0 (x = 0 - 3)** or **GxASCTRL (x = 0 - 3)**) for the intended source of equidistant conversions. In timer mode, a request of this source is triggered and arbitrated, but only started when the trigger signal is removed (see **Figure 27-28**) and the converter is idle.

To ensure that the converter is idle and the start of conversion can be controlled by the trigger signal, the equidistant conversion requests must receive highest priority. The preface time between request trigger and conversion start must be long enough for a currently active conversion to finish.

The frequency of signal REQTRx defines the sampling rate and its high time defines the preface time interval where the corresponding request source takes part in the arbitration.

Depending on the used request source, equidistant sampling is also supported for a sequence of channels. It is also possible to do equidistant sampling for more than one request source in parallel if the preface times and the equidistant conversions do not overlap.

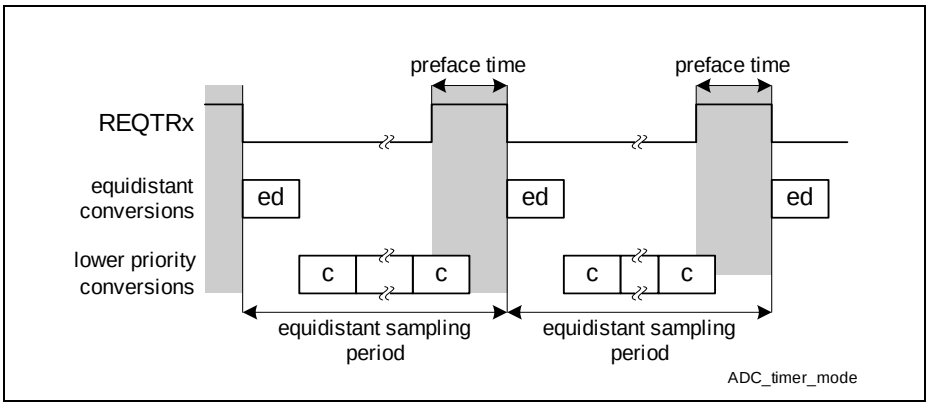


Figure 27-28 Timer Mode for Equidistant Sampling

27.11 Safety Features

Several test features can be enabled to verify the validity of the analog input signals of an application. These test features aim at different sections of the signal flow:

- **Broken Wire Detection** validates the connection from the sensor to the input pin
- **Signal Path Test Modes** validate the operation of the internal analog input multiplexer and the Analog/Digital converter itself
- **Automatic Execution of Test Sequences** supports the safety software by generating predefined sets of data to validate the signal path

27.11.1 Broken Wire Detection

To test the proper connection of an external analog sensor to its input pin, the converter's capacitor can be precharged to a selectable value before the regular sample phase. If the connection to the sensor is interrupted the subsequent conversion value will rather represent the precharged value than the expected sensor result. By using a precharge voltage outside the expected result range (broken wire detection preferably uses V_{AGND} and/or V_{AREF}) a valid measurement (sensor connected) can be distinguished from a failure (sensor detached).

While broken wire detection is disabled, the converter's capacitor is precharged to $V_{AREF}/2$.

Note: The duration of the complete conversion is increased by the preparation phase (same as the sample phase) if the broken wire detection is enabled. This influences the timing of conversion sequences.

Broken wire detection can be enabled for each channel separately by bitfield BWDEN in the corresponding channel control register (GxCHCTRy). Bitfield BWDCH selects the level for the preparation phase.

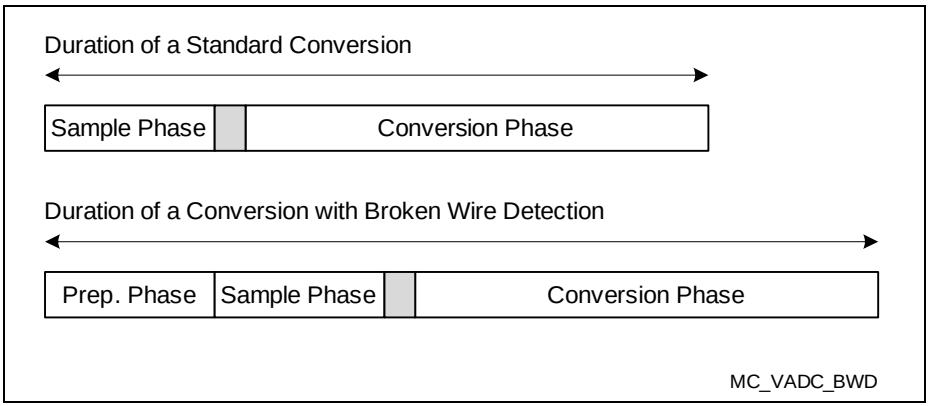


Figure 27-29 Broken Wire Detection

27.11.2 Signal Path Test Modes

Additional test structures can be activated to test the signal path from the sensor to the input pin and the internal signal path from the input pin through the multiplexer to the converter. These test structures apply additional loads to the signal path (see summary in **Figure 27-30**).

Multiplexer Diagnostics

To test the proper operation of the internal analog input multiplexer, additional pull-up and/or pull-down devices can be connected to channels CH1 and CH2. In combination with a known external input signal this test function shows if the multiplexer connects any pin to the converter input and if this is the correct pin.

Pull-Down Diagnostics

One single input channel provides a further strong pull-down (R_{PDD}) that can be activated to verify the external connection to a sensor.

Converter Diagnostics

To test the proper operation of the converter itself, several signals can be connected to the converter input. The test signals can be connected to the converter input either instead of the standard input signal or in parallel to the standard input signal.

The test signal can be selected from four different signals as shown in **Figure 27-30**.

Attention: If the Multiplexer Diagnostics are used on channels that are overlaid on IO ports, enable these functions by setting the corresponding control bits in registers Pxx_PCSR.

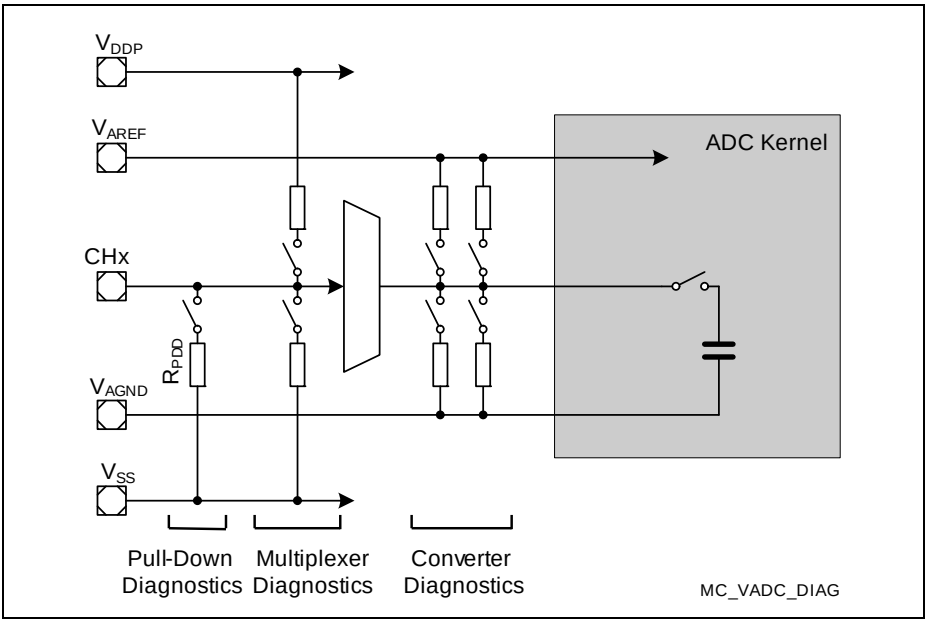


Figure 27-30 Signal Path Test

Note: When internal pull devices are activated, they may need some time to pull the input signal to the intended level, depending on the connected external circuitry. This can be accomplished by increasing the sample time for those conversions.

27.11.3 Configuration of Test Functions

The pull-up and pull-down devices for the test functions can be enabled individually under software control. Various test levels can be applied controlling the devices in an adequate way. Because these test functions interfere with the normal operation of the A/D Converters, they are controlled by a separate register set.

Not all test options are available for each channel. Selecting an unavailable function has no effect.

*Note: The pull device control bits in register **GLOBTF** can only activate the test pull devices of a group, when the corresponding control bit in register **GLOBTE** is zero.*

GLOBTF
Global Test Functions Register (0160_H) Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RC WC	**RC EN**	**0**	**0**	**0**	**0**	**0**	**0**	**MD WC**	**0**	**0**	**0**	**0**	**MD PU**	**MD PD**	**PDD**
w	rw	r	r	r	r	r	r	w	r	r	r	r	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CD WC	**0**	**0**	**0**	**0**	**CD SEL**		**CD EN**	**CDGR**				**CDCH**			
w	r	r	r	r	rw		rw	rw				rw			

Field	Bits	Type	Description
CDCH	[3:0]	rw	**Conversion Diagnostics Channel** Defines the channel number to be used for diagnostic conversions.
CDGR	[7:4]	rw	**Conversion Diagnostics Group** Defines the group number to be used for diagnostic conversions.
CDEN	8	rw	**Converter Diagnostics Enable** 0_B All diagnostic pull devices are disconnected 1_B Diagnostic pull devices connected as selected by bitfield CDSEL

Field	Bits	Type	Description
CDSEL	[10:9]	rw	**Converter Diagnostics Pull-Devices Select** 00_B Connected to VAREF 01_B Connected to VAGND 10_B Connected to 1/3rd VAREF 11_B Connected to 2/3rd VAREF
0	[14:11]	r	**Reserved, write 0, read as 0**
CDWC	15	w	**Write Control for Conversion Diagnostics** 0_B No write access to parameters 1_B Bitfields CDSEL, CDEN, CDGR, CDCH can be written
PDD	16	rw	**Pull-Down Diagnostics Enable** 0_B Disconnected 1_B The pull-down diagnostics device is active *Note: Channels with pull-down diagnostics device are marked in **Table 27-12**.*
MDPD, MDPU	17, 18	rw	**Multiplexer Diagnostics Pull-Devices Enable** 0_B Disconnected 1_B The respective device is active Connecting combinations of pull-up and/or pull-down devices generate various loads for testing. *Note: Channels with pull-down diagnostics device are marked in **Table 27-12**.*
0	[22:19]	r	**Reserved, write 0, read as 0**
MDWC	23	w	**Write Control for Multiplexer Diagnostics** 0_B No write access to parameters 1_B Bitfields MDPD, MDPU, PDD can be written
0	[29:24]	r	**Reserved, write 0, read as 0**
RCEN	30	rw	**Reference Channel Enable** 0_B No reference signals available 1_B Internal reference signals can be converted
RCWC	31	w	**Write Control for Reference Control** 0_B No write access to parameters 1_B Bitfield RCEN can be written

When automatic test sequences (see **Section 27.11.4**) are enabled, the scheduled conversions will activate the configured test pull devices. To make sure that this does not happen inadvertently, the activation of the test functions must be particularly enabled.

The release is done by setting the corresponding bit in the Global Test Enable register (GLOBTE). This prevents spurious control signals from spoiling a configured test sequence. GLOBTE can be write-protected via bit APTF in register ACCPROT1 (ACCPROT1 itself is safe-endinit protected).

*Note: The pull device control bits in register **GLOBTF** can only activate the test pull devices of a group, when the corresponding control bit in register **GLOBTE** is zero.*

The Global Test Enable register enables the control of the available test pull devices for each group separately.

GLOBTE
Global Test Enable Register (0164$_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**TFEG1**	**TFEG0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw

Field	Bits	Type	Description
TFEG0, TFEG1	0, 1	rw	**Test Function Enable for Group x** Enables the activation of test pull devices by test sequences of group x. 0_B No test functions for group x sequences 1_B Test functions can be activated by group x sequences
0	[31:2]	r	**Reserved, write 0, read as 0**

27.11.4 Automatic Execution of Test Sequences

Validating the correct connection of an input signal (sensor) to the converter in a system requires some action, depending on the properties of the signal to be converted. For example, sensors with a limited signal range can be validated by checking the conversion result for values outside of the defined signal range.

In other cases pull devices (up / down) can be connected to the signal input, which alter the signal level and, as a consequence, the conversion result in a known way. The alteration can be anticipated when the signal source impedance is known.

Timing Considerations

Test sequences that are executed during the initialization of a system usually are not time-critical. They increase the startup time but otherwise have no impact on the application. Safety-critical systems, however, monitor the operability of the system during the operation time. In this case, the test sequences must be interleaved with the normal application sequences. A test sequence shall be executed immediately after an application sequence, so the relaxation time until the next application sequence becomes maximal.

Test sequences can be triggered by each available request source.

In either case the overall timing must be selected so overlaps are avoided.

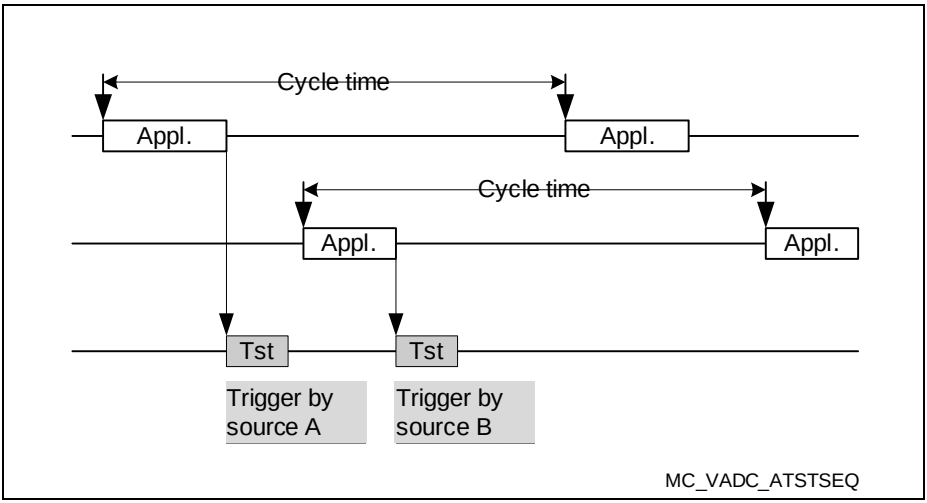


Figure 27-31 Positions of Test Sequences

Note that the test conversions can target arbitrary channels.

Using the Test Queue

The test queue request source (queue 3) is similar to the standard queued request source (queue 0) with some additional control bits to select test functions of the pins.

Leaving these control bits as zero allows using queue 3 as an additional request source for the application.

Note: When queue 3 is to be used, the arbiter must be programmed to an 8-step arbitration cycle, so the queue can take part in the arbitration process.

Queue 3 can be activated by external triggers (same as other sources) or by internal triggers from the other group request sources. These additional internal trigger signals are selected by an additional multiplexer whose output is connected to the uppermost input of the gate multiplexer. To use the internal triggers, select the uppermost trigger input (XTSEL = 1111_B) and the uppermost gate input (GTSEL = 1111_B).

A trigger counter activates queue 3 after a defined number of triggers. Test sequences can, therefore, be inserted into the application-specific conversion sequences at a certain rate without additional software intervention.

Control the internal triggers and the trigger counter via register **GxTRCTR (x = 0 - 3)**.

Queue 3 contains entries building a sequence of test conversions which enable arbitrary diagnostic functions. A trigger for a test sequence is generated automatically when the sequence of a standard request source is completed.

The trigger sequence counter starts a (test) sequence after a programmable number of triggers. The (test) sequence is started if the trigger sequence counter equals 0 when the trigger occurs. For other values of the trigger sequence counter, each trigger decrements the counter. No software intervention is required to interleave test sequences into the normal operation.

Activation/Deactivation of Pull-Devices

When specific test functions are enabled for an entry in queue 3, the correponding enabled pull-devices are controlled automatically.
They are activated when the sample phase for this conversion begins.
They are deactivated automatically after the sample phase is completed.

Depending on the properties of the connected sensor, a test sequence may require an additional settling time. This can be covered by selecting an increased sample time for these test conversions.

Detecting Irregular Test Sequences

An overflow bit (OV) is set when an internal trigger for queue 3 occurs while the queue is active. This indicates that a test sequence has been triggered when it cannot be executed as expected.

Queue 3 is assumed to be active between the occurrence of an internal trigger and the generation of a source event of queue 3. This phase is indicated by bit Q3ACT = 1.

Software can clear bits OV and Q3ACT by writing 1 to bit COV (see register **GxTRCTR (x = 0 - 3)**).

Securing the Test Functions

When test sequences are enabled, the scheduled conversions will activate the configured test pull devices. To make sure that this does not happen inadvertently, the activation of the test functions must be explicitly enabled.

The release is done by setting the corresponding bit in the Global Test Enable register (**GLOBTE**). This prevents spurious control signals from spoiling a configured test sequence. GLOBTE can be write-protected via bit APTF in register ACCPROT1 (ACCPROT1 itself is safe-endinit protected).

*Note: The pull device control bits in register **GLOBTF** can only activate the test pull devices of a group, when the corresponding control bit in register **GLOBTE** is zero.*

Evaluation of Test Results

The effect of the selected test pull devices depends on the properties of the connected signal source. These properties are system-specific and are not known to the controller. Therefore, the system software must evaluate the test results by comparing them to the expected results. This is influenced by several parameters:

- Selected pull device
- Properties of signal source, e.g. impedance
- Properties of input signal (e.g. current level of dynamic signals)
- Acceptable failure detection time

These aspects must be evaluated to assess the validity of the input signal on the channel being checked.

27.12 External Multiplexer Control

The number of analog input channels can be increased by connecting external analog multiplexers to an input MUX channel. The ADC can be configured to control these external multiplexers automatically.

For each available EMUX interface (see register **EMUXSEL**) one MUX channel or a set of MUX channels from one group can be selected for this operating mode. The ADC supports 1-out-of-8 multiplexers with several control options:

- **Sequence mode** automatically converts all configured external channels when the selected MUX channel is encountered. In the example in **Figure 27-32** the following conversions are done: --4-32-31-30-2-1-0--4-32-31-30-2-1-0--…
- **Single-step mode** converts one external channel of the configured sequence when the selected MUX channel is encountered. In the example in **Figure 27-32** the following conversions are done: --4-32-2-1-0--4-31-2-1-0--4-30-2-1-0--4-32-… (Single-step mode works best with one channel)
- **Steady mode** converts the configured external channel when the selected MUX channel is encountered. In the example in **Figure 27-32** the following conversions are done: --4-32-2-1-0--4-32-2-1-0--4-32-2-1-0--…

*Note: The example in **Figure 27-32** has an external multiplexer connected to MUX channel CH3. The start selection value EMUXSET is assumed as 2.*

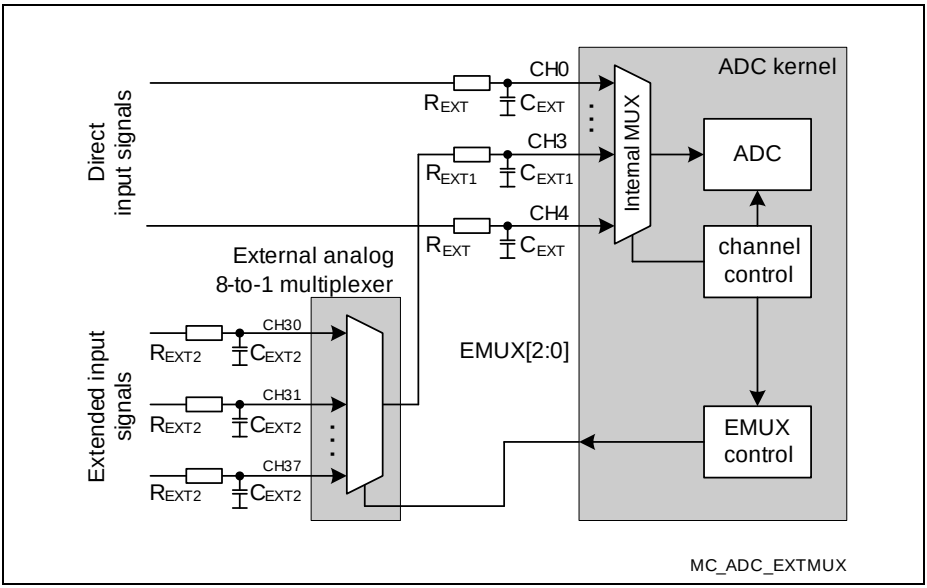


Figure 27-32 External Analog Multiplexer Example

Bitfield EMUXACT determines the control information sent to the external multiplexer. In single-step mode, EMUXACT is updated after each conversion of an enabled MUX channel. If EMUXACT = 000_B it is reloaded from bitfield EMUXSET, otherwise it is decremented by 1.

Additional external channels may have different properties due to the modified signal path. Local filters may be used at the additional inputs (R_{EXT2}-C_{EXT2} on CH3x in **Figure 27-32**). For applications where the external multiplexer is located far from the ADC analog input, it is recommended to add an RC filter directly at the analog input of the ADC (R_{EXT1}-C_{EXT1} on CH3 in **Figure 27-32**).

Note: Each RC filter limits the bandwidth of the analog input signal.

Conversions for external channels, therefore, use the alternate conversion mode setting CME. This automatically selects a different conversion mode if required.

Switching the external multiplexer usually requires an additional settling time for the input signal. Therefore, the alternate sample time setting STCE is applied each time the external channel is changed. This automatically fulfills the different sampling time requirements in this case.

Control Signals

The external channel number that controls the external multiplexer can be output in standard binary format or Gray-coded. Gray code avoids intermediate multiplexer switching when selecting a sequence of channels, because only one bit changes at a time. **Table 27-7** indicates the resulting codes.

Table 27-7 EMUX Control Signal Coding

Channel	0	1	2	3	4	5	6	7
Binary	000_B	001_B	010_B	011_B	100_B	101_B	110_B	111_B
Gray	000	001	011	010	110	111	101	100

Operation Without External Multiplexer

If no external multiplexers are used in an application, the reset values of the control registers provide the appropriate setup.

EMUXMODE = 00_B disables the automatic EMUX control.

Since the control output signals are alternate port output signals, they are only visible at the respective pins if explicitly selected.

In each group an arbitrary MUX channel or set of MUX channels can be assigned to external multiplexer control (register **GxEMUXCTR (x = 0 - 3)**). Each available port interface selects the group whose control lines are output (register **EMUXSEL**).

GxEMUXCTR (x = 0 - 3)
External Multiplexer Control Register, Group x
($x * 0400_H + 05F0_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EMX WC	EMX CSS	EMX ST	EMX COD	EMUX MODE		EMUX CH									
w	rw	rw	rw	rw		rw									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	EMUX ACT			0	0	0	0	0	EMUX SET		
r	r	r	r	r	rh			r	r	r	r	r	rw		

Field	Bits	Type	Description
EMUXSET	[2:0]	rw	**External Multiplexer Start Selection**[1)] Defines the initial setting for the external multiplexer.
0	[7:3]	r	**Reserved, write 0, read as 0**
EMUXACT	[10:8]	rh	**External Multiplexer Actual Selection** Defines the current value for the external multiplexer selection. This bitfield is loaded from bitfield EMUXSET and modified according to the operating mode selected by bitfield EMUXMODE.
0	[15:11]	r	**Reserved, write 0, read as 0**
EMUXCH	[25:16]	rw	**External Multiplexer Channel Select** Defines the channel(s) to which the external multiplexer control is applied. **EMXCSS = 0: Channel number**, the lower 5 bits select an arbitrary channel (valid numbers are limited by the number of available channels, unused bits shall be 0) **EMXCSS = 1: Channel enable**, each bit enables the associated channel (multiple channels can be selected/enabled)

Field	Bits	Type	Description
EMUXMODE	[27:26]	rw	**External Multiplexer Mode** 00_B Software control (no hardware action) 01_B Steady mode (use EMUXSET value) 10_B Single-step mode[1)2)] 11_B Sequence mode[1)]
EMXCOD	28	rw	**External Multiplexer Coding Scheme** 0_B Output the channel number in binary code 1_B Output the channel number in Gray code
EMXST	29	rw	**External Multiplexer Sample Time Control** 0_B Use STCE whenever the setting changes 1_B Use STCE for each conversion of an external channel
EMXCSS	30	r	**External Multiplexer Channel Selection Style** 0_B Channel number: Bitfield EMUXCH selects an arbitrary channel 1_B Channel enable: Each bit of bitfield EMUXCH selects the associated channel for EMUX control
EMXWC	31	w	**Write Control for EMUX Configuration** 0_B No write access to EMUX cfg. 1_B Bitfields EMXMODE, EMXCOD, EMXST, EMXCSS can be written

1) For single-step mode and sequence mode: Select the start value before selecting the respective mode.

2) Single-step mode modifies the EMUX channel number each time an EMUX-enabled channel is converted. Therefore, single-step mode works best with a single channel, because otherwise some external channels may be skipped.

Register EMUXSEL is a global register which assigns an arbitrary group to each of the EMUX interfaces.

EMUXSEL
External Multiplexer Select Register

($03F0_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**EMUX GRP1**				**EMUX GRP0**			
r	r	r	r	r	r	r	r	rw				rw			

Field	Bits	Type	Description
EMUXGRP0, EMUXGRP1	[3:0], [7:4]	rw	**External Multiplexer Group for Interface x** Defines the group whose external multiplexer control signals are routed to EMUX interface x.[1)]
0	[31:8]	r	**Reserved, write 0, read as 0**

1) The pins that are associated with each EMUX interface are listed in **Table 27-13 "Digital Connections in the TC21x/TC22x/TC23x" on Page 27-176**.

27.13 Service Request Generation

Each A/D Converter can activate up to 4 group-specific service request output signals and up to 4 shared service request output signals to issue an interrupt or to trigger a DMA channel. Two common service request groups are available, see **Table 27-8 "General Converter Configuration in the TC21x/TC22x/TC23x" on Page 27-167**.

Several events can be assigned to each service request output. Service requests can be generated by three types of events:

- **Request source events:** indicate that a request source completed the requested conversion sequence and the application software can initiate further actions.
  For a scan source (group or background), the event is generated when the complete defined set of channels (pending bits) has been converted.
  For a group queue source, the event is generated according to the programming, i.e. when a channel with enabled source interrupt has been converted or when an invalid entry is encountered.
- **Channel events:** indicate that a conversion is finished. Optionally, channel events can be restricted to result values within a programmable value range. This offloads the CPU/DMA from background tasks, i.e. a service request is only activated if the specified conversion result range is met or exceeded.
- **Result events:** indicate a new valid result in a result register. Usually, this triggers a read action by the CPU (or DMA). Optionally, result events can be generated only at a reduced rate if data reduction is active.
  For example, a single DMA channel can read the results for a complete auto-scan sequence, if all channels of the sequence target the same result register and the transfers are triggered by result events.

Each ADC event is indicated by a dedicated flag that can be cleared by software. If a service request is enabled for a certain event, the service request is generated for each event, independent of the status of the corresponding event indication flag. This ensures efficient DMA handling of ADC events (the ADC event can generate a service request without the need to clear the indication flag).

Event flag registers indicate all types of events that occur during the ADC's operation. Software can set each flag by writing a 1 to the respective position in register GxCEFLAG/GxRFLAG to trigger an event. Software can clear each flag by writing a 1 to the respective position in register GxCEFCLR/GxREFCLR. If enabled, service requests are generated for each occurrance of an event, even if the associated flag remains set.

GxSEFLAG (x = 0 - 3)
Source Event Flag Register, Group x

(x * 0400_H + 0588_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**SEV 3**	**0**	**SEV 1**	**SEV 0**
r	r	r	r	r	r	r	r	r	r	r	r	rwh	r	rwh	rwh

Field	Bits	Type	Description
SEV0, **SEV1,** **SEV3**	0, 1, 3	rwh	**Source Event 0/1/3** 0_B No source event 1_B A source event has occurred
0	[31:2]	r	**Reserved, write 0, read as 0**

Note: Software can set all flags in register GxSEFLAG and trigger the corresponding event by writing 1 to the respective bit. Writing 0 has no effect.
Software can clear all flags in register GxSEFLAG by writing 1 to the respective bit in register GxSEFCLR.

GxCEFLAG (x = 0 - 3)
Channel Event Flag Register, Group x

(x * 0400_H + 0580_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEV 15	**CEV 14**	**CEV 13**	**CEV 12**	**CEV 11**	**CEV 10**	**CEV 9**	**CEV 8**	**CEV 7**	**CEV 6**	**CEV 5**	**CEV 4**	**CEV 3**	**CEV 2**	**CEV 1**	**CEV 0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
CEVy (y = 0 - 15)	y	rwh	**Channel Event for Channel y** 0_B No channel event 1_B A channel event has occurred
0	[31:16]	r	**Reserved, write 0, read as 0**

Note: Software can set all flags in register GxCEFLAG and trigger the corresponding event by writing 1 to the respective bit. Writing 0 has no effect.
Software can clear all flags in register GxCEFLAG by writing 1 to the respective bit in register GxCEFCLR.

GxREFLAG (x = 0 - 3)
Result Event Flag Register, Group x

(x * 0400_H + 0584_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REV 15	**REV 14**	**REV 13**	**REV 12**	**REV 11**	**REV 10**	**REV 9**	**REV 8**	**REV 7**	**REV 6**	**REV 5**	**REV 4**	**REV 3**	**REV 2**	**REV 1**	**REV 0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
REVy (y = 0 - 15)	y	rwh	**Result Event for Result Register y** 0_B No result event 1_B New result was stored in register GxRESy
0	[31:16]	r	**Reserved, write 0, read as 0**

Note: Software can set all flags in register GxREFLAG and trigger the corresponding event by writing 1 to the respective bit. Writing 0 has no effect.
Software can clear all flags in register GxREFLAG by writing 1 to the respective bit in register GxREFCLR.

GxSEFCLR (x = 0 - 3)

Source Event Flag Clear Register, Group x

($x * 0400_H + 0598_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**SEV 3**	**0**	**SEV 1**	**SEV 0**
r	r	r	r	r	r	r	r	r	r	r	r	w	r	w	w

Field	Bits	Type	Description
SEV0, SEV1, SEV3	0, 1, 3	w	**Clear Source Event 0/1/3** 0_B No action 1_B Clear the source event flag in GxSEFLAG
0	2, [31:4]	r	**Reserved, write 0, read as 0**

GxCEFCLR (x = 0 - 3)

Channel Event Flag Clear Register, Group x

($x * 0400_H + 0590_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEV 15	**CEV 14**	**CEV 13**	**CEV 12**	**CEV 11**	**CEV 10**	**CEV 9**	**CEV 8**	**CEV 7**	**CEV 6**	**CEV 5**	**CEV 4**	**CEV 3**	**CEV 2**	**CEV 1**	**CEV 0**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
CEVy (y = 0 - 15)	y	w	**Clear Channel Event for Channel y** 0_B No action 1_B Clear the channel event flag in GxCEFLAG
0	[31:16]	r	**Reserved, write 0, read as 0**

GxREFCLR (x = 0 - 3)
Result Event Flag Clear Register, Group x

(x * 0400_H + 0594_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REV 15	**REV 14**	**REV 13**	**REV 12**	**REV 11**	**REV 10**	**REV 9**	**REV 8**	**REV 7**	**REV 6**	**REV 5**	**REV 4**	**REV 3**	**REV 2**	**REV 1**	**REV 0**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
REVy (y = 0 - 15)	y	w	**Clear Result Event for Result Register y** 0_B No action 1_B Clear the result event flag in GxREFLAG
0	[31:16]	r	**Reserved, write 0, read as 0**

GLOBEFLAG
Global Event Flag Register (00E0$_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**REVGLBCLR**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**SEVGLBCLR**
r	r	r	r	r	r	r	w	r	r	r	r	r	r	r	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**REVGLB**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**SEVGLB**
r	r	r	r	r	r	r	rwh	r	r	r	r	r	r	r	rwh

Field	Bits	Type	Description
SEVGLB	0	rwh	**Source Event (Background)** 0_B No source event 1_B A source event has occurred
0	[7:1]	r	**Reserved, write 0, read as 0**
REVGLB	8	rwh	**Global Result Event** 0_B No result event 1_B New result was stored in register GLOBRES
0	[15:9]	r	**Reserved, write 0, read as 0**
SEVGLBCLR	16	w	**Clear Source Event (Background)** 0_B No action 1_B Clear the source event flag SEVGLB
0	[23:17]	r	**Reserved, write 0, read as 0**
REVGLBCLR	24	w	**Clear Global Result Event** 0_B No action 1_B Clear the result event flag REVGLB
0	[31:25]	r	**Reserved, write 0, read as 0**

Note: Software can set flags REVGLB and SEVGLB and trigger the corresponding event by writing 1 to the respective bit. Writing 0 has no effect.
Software can clear these flags by writing 1 to bit REVGLBCLR and SEVGLBCLR, respectively.
Setting both bits (e.g. SEVGLB and SEVGLBCLR) simultaneously clears the corresponding flag (e.g. SEVGLB).

Node Pointer Registers

Requests from each event source can be directed to a set of service request nodes via associated node pointers. Requests from several sources can be directed to the same node; in this case, they are ORed to the service request output signal.

GxSEVNP (x = 0 - 3)
Source Event Node Pointer Register, Group x
(x * 0400_H + $05C0_H$) **Reset Value: $0000\ 0000_H$**

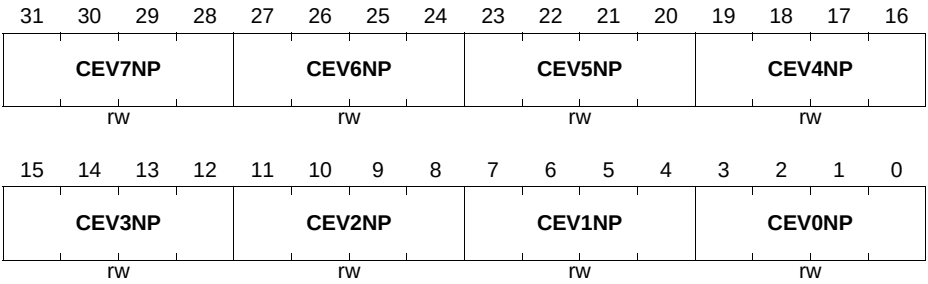
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEV3NP				**0**	**0**	**0**	**0**	**SEV1NP**				**SEV0NP**			
rw				r	r	r	r	rw				rw			

Field	Bits	Type	Description
SEV0NP, SEV1NP, SEV3NP	[3:0], [7:4], [15:12]	rw	**Service Request Node Pointer Source Event i[1)]** Routes the corresponding event trigger to one of the service request lines (nodes). 0000_B Select service request line 0 of group x … 0011_B Select service request line 3 of group x 0100_B Select shared service request line 0 … 0111_B Select shared service request line 3 $1xxx_B$ Reserved *Note: For shared service request lines see common groups in **Table 27-8**.*
0	[11:8], [31:16]	r	**Reserved, write 0, read as 0**

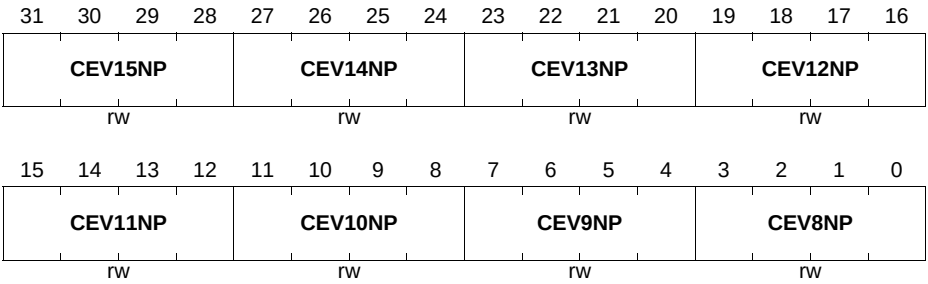
1) Source 0 is an 8-stage queued source, source 1 is a channel scan source, source 3 is an 8-stage queued source.

GxCEVNP0 (x = 0 - 3)
Channel Event Node Pointer Register 0, Group x
(x * 0400_H + $05A0_H$) **Reset Value: $0000\ 0000_H$**



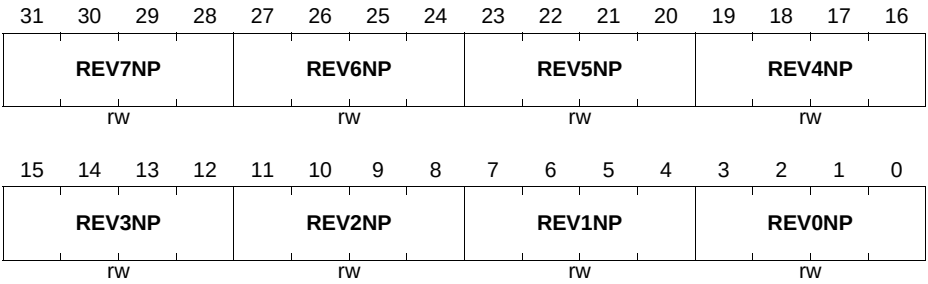
Field	Bits	Type	Description
CEV0NP, **CEV1NP,** **CEV2NP,** **CEV3NP,** **CEV4NP,** **CEV5NP,** **CEV6NP,** **CEV7NP**	[3:0], [7:4], [11:8], [15:12], [19:16], [23:20], [27:24], [31:28]	rw	**Service Request Node Pointer Channel Event i** Routes the corresponding event trigger to one of the service request lines (nodes). 0000_B Select service request line 0 of group x … 0011_B Select service request line 3 of group x 0100_B Select shared service request line 0 … 0111_B Select shared service request line 3 $1xxx_B$ Reserved *Note: For shared service request lines see common groups in **Table 27-8**.*

GxCEVNP1 (x = 0 - 3)
Channel Event Node Pointer Register 1, Group x
(x * 0400_H + $05A4_H$) **Reset Value: $0000\ 0000_H$**



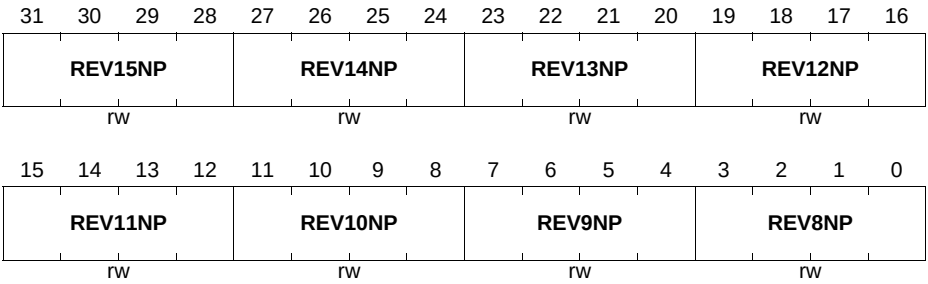
Field	Bits	Type	Description
CEV8NP, **CEV9NP,** **CEV10NP,** **CEV11NP,** **CEV12NP,** **CEV13NP,** **CEV14NP,** **CEV15NP**	[3:0], [7:4], [11:8], [15:12], [19:16], [23:20], [27:24], [31:28]	rw	**Service Request Node Pointer Channel Event i** Routes the corresponding event trigger to one of the service request lines (nodes). 0000_B Select service request line 0 of group x … 0011_B Select service request line 3 of group x 0100_B Select shared service request line 0 … 0111_B Select shared service request line 3 $1xxx_B$ Reserved *Note: For shared service request lines see common groups in **Table 27-8**.*

GxREVNP0 (x = 0 - 3)
Result Event Node Pointer Register 0, Group x
(x * 0400_H + $05B0_H$) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
REV0NP, **REV1NP,** **REV2NP,** **REV3NP,** **REV4NP,** **REV5NP,** **REV6NP,** **REV7NP**	[3:0], [7:4], [11:8], [15:12], [19:16], [23:20], [27:24], [31:28]	rw	**Service Request Node Pointer Result Event i** Routes the corresponding event trigger to one of the service request lines (nodes). 0000_B Select service request line 0 of group x … 0011_B Select service request line 3 of group x 0100_B Select shared service request line 0 … 0111_B Select shared service request line 3 $1xxx_B$ Reserved *Note: For shared service request lines see common groups in **Table 27-8**.*

GxREVNP1 (x = 0 - 3)
Result Event Node Pointer Register 1, Group x
(x * 0400_H + $05B4_H$) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
REV8NP, **REV9NP,** **REV10NP,** **REV11NP,** **REV12NP,** **REV13NP,** **REV14NP,** **REV15NP**	[3:0], [7:4], [11:8], [15:12], [19:16], [23:20], [27:24], [31:28]	rw	**Service Request Node Pointer Result Event i** Routes the corresponding event trigger to one of the service request lines (nodes). 0000_B Select service request line 0 of group x … 0011_B Select service request line 3 of group x 0100_B Select shared service request line 0 … 0111_B Select shared service request line 3 $1xxx_B$ Reserved *Note: For shared service request lines see common groups in **Table 27-8**.*

GLOBEVNP
Global Event Node Pointer Register

(0140_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**REV0NP**			
r	r	r	r	r	r	r	r	r	r	r	r	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**SEV0NP**			
r	r	r	r	r	r	r	r	r	r	r	r	rw			

Field	Bits	Type	Description
SEV0NP	[3:0]	rw	**Service Request Node Pointer Backgr. Source** Routes the corresponding event trigger to one of the service request lines (nodes). 0000_B Select shared service request line 0 of common service request group 0 … 0011_B Select shared service request line 3 of common service request group 0 0100_B Select shared service request line 0 of common service request group 1 … 0111_B Select shared service request line 3 of common service request group 1 $1xxx_B$ Reserved *Note: For shared service request lines see common groups in **Table 27-8**.*
0	[15:4]	r	**Reserved, write 0, read as 0**

Field	Bits	Type	Description
REV0NP	[19:16]	rw	**Service Request Node Pointer Global Result** Routes the corresponding event trigger to one of the service request lines (nodes). 0000_B Select shared service request line 0 of common service request group 0 … 0011_B Select shared service request line 3 of common service request group 0 0100_B Select shared service request line 0 of common service request group 1 … 0111_B Select shared service request line 3 of common service request group 1 $1xxx_B$ Reserved *Note: For shared service request lines see common groups in **Table 27-8**.*
0	[31:20]	r	**Reserved, write 0, read as 0**

Software Service Request Activation

Each service request can be activated via software by setting the corresponding bit in register **GxSRACT (x = 0 - 3)**. This can be used for evaluation and testing purposes.

*Note: For shared service request lines see common groups in **Table 27-8**.*

GxSRACT (x = 0 - 3)
Service Request Software Activation Trigger, Group x
(x * 0400_H + $05C8_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**ASSR3**	**ASSR2**	**ASSR1**	**ASSR0**	**0**	**0**	**0**	**0**	**AGSR3**	**AGSR2**	**AGSR1**	**AGSR0**
r	r	r	r	w	w	w	w	r	r	r	r	w	w	w	w

Field	Bits	Type	Description
AGSRy (y = 0 - 3)	y	w	**Activate Group Service Request Node y** 0_B No action 1_B Activate the associated service request line
0	[7:4]	r	**Reserved, write 0, read as 0**
ASSRy (y = 0 - 3)	8 + y	w	**Activate Shared Service Request Node y** 0_B No action 1_B Activate the associated service request line
0	[31:12]	r	**Reserved, write 0, read as 0**

27.14 Implementation into the TC21x/TC22x/TC23x

This section describes the actual implementation of the ADC module into the TC21x/TC22x/TC23x, i.e. the incorporation into the microcontroller system.

The following information is listed:

- **"Product-Specific Configuration" on Page 27-167**
- **"Summary of Registers and Locations" on Page 27-169**
- **"Analog Module Connections in the TC21x/TC22x/TC23x" on Page 27-174**
- **"Digital Module Connections in the TC21x/TC22x/TC23x" on Page 27-176**

27.14.1 Product-Specific Configuration

The functional description describes the features and operating modes of the A/D Converters in a general way. This section summarizes the configuration that is available in this product (TC21x/TC22x/TC23x).

The standard product device features 2 converter groups (G0, G1) which provide 14 analog input channels each. 12 channels can be selected directly, 2 additional channels (for internal test signals) can be selected via the alias feature.

The special ADAS device features 2 additional converter groups (G0, G1, G2, G3) which provide one analog input channel each. These additional channels are overlaid on existing channels of the standard device.

Note: The additional ADAS groups do not provide the test request source queue 3.

Each converter group is equipped with a separate analog converter module and a dedicated analog input multiplexer.

Table 27-8 General Converter Configuration in the TC21x/TC22x/TC23x

Converter Group	Input Channels	Channels w. Altern. Reference	12-bit Performance	Common Service Req. Group	Associated Standard Reference
G0	0 … 11[1)]	8	Calibrated	C0	V_{AREF}, V_{AGND}
G1	0 … 11[1)]	12	Calibrated	C1	V_{AREF}, V_{AGND}
G2	0	0	Calibrated	C0	V_{AREF}, V_{AGND}
G3	0	0	Calibrated	C1	V_{AREF}, V_{AGND}

1) Two additional channels for internal test signals via the alias feature.

Note: The standard device provides groups G0 … G1,
the ADAS device provides groups G0 … G3.

Synchronization Groups in the TC21x/TC22x/TC23x

The converter kernels in the TC21x/TC22x/TC23x can be connected to synchronization groups to achieve parallel conversion of several input channels.

Table 27-9 summarizes which kernels can be synchronized for parallel conversions.

Table 27-9 Synchronization Groups in the TC21x/TC22x/TC23x

ADC Kernel	Synchr. Group	Master selected by control input CIx[1]			
		CI0[2]	CI1	CI2	CI3
ADC00	A	ADC00	ADC01	ADC02	ADC03
ADC01	A	ADC01	ADC00	ADC02	ADC03
ADC02	A	ADC02	ADC00	ADC01	ADC03
ADC03	A	ADC03	ADC00	ADC01	ADC02

1) The control input is selected by bitfield STSEL in register **GxSYNCTR (x = 0 - 3)**. Select the corresponding ready inputs accordingly by bits EVALRx.

2) Control input CI0 always selects the own control signals of the corresponding ADC kernel. This selection is meant for the synchronization master or for stand-alone operation.

Note: The shaded part of the above table applies to the ADAS device only.

27.14.2 Summary of Registers and Locations

The Versatile ADC is built from a series of converter blocks that are controlled in an identical way. This makes programming versatile and scalable. The corresponding registers, therefore, have an individual offset assigned (see **Table 27-11**). The exact register location is obtained by adding the respective register offset to the base address (see **Table 27-10**) of the corresponding group.

Due to the regular group structure, several registers appear within each group. Other registers are provided for each channel. This is indicated in the register overview table by placeholders:

- X###$_H$ means: x × 0400$_H$ + 0###$_H$, for x = 0 - 3
- ###Y$_H$ means: ###0$_H$ + y × 0004$_H$, for y = 0 - N (depends on register type)

Table 27-10 Registers Address Space

Module	Base Address	End Address	Note
VADC	F002 0000$_H$	F002 3FFF$_H$	

Table 27-11 Registers Overview

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Page Num.
			Read	Write	
ID	Module Identification Register	0008$_H$	U, SV	BE	27-14
CLC	Clock Control Register	0000$_H$	U, SV	SV E, P	27-15
OCS	OCDS Control and Status Register	0028$_H$	U, SV	SV P	27-16
ACCEN0	Access Enable Register 0	003C$_H$	U, SV	SV SE	27-18
KRST0	Kernel Reset Register 0	0034$_H$	U, SV	SV E, P	27-19
KRST1	Kernel Reset Register 1	0030$_H$	U, SV	SV E, P	27-20
KRSTCLR	Kernel Reset Clear Register	002C$_H$	U, SV	SV E, P	27-20
GLOBCFG	Global Configuration Register	0080$_H$	U, SV	U, SV P	27-23
ACCPROT0	Access Protection Register 0	0088$_H$	U, SV	SV, SE, P	27-27

Table 27-11 Registers Overview (cont'd)

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Page Num.
			Read	Write	
ACCPROT1	Access Protection Register 1	$008C_H$	U, SV	SV, SE, P	27-28
GxARBCFG	Arbitration Configuration Register	$X480_H$	U, SV	U, SV P	27-83
GxARBPR	Arbitration Priority Register	$X484_H$	U, SV	U, SV P	27-86
GxCHASS	Channel Assignment Register, Group x	$X488_H$	U, SV	U, SV P	27-25
GxRRASS	Result Assignment Register, Group x	$X48C_H$	U, SV	U, SV P	27-26
GxQCTRL0	Queue 0 Source Control Register, Group x	$X500_H$	U, SV	U, SV P	27-38
GxQMR0	Queue 0 Mode Register, Group x	$X504_H$	U, SV	U, SV P	27-40
GxQSR0	Queue 0 Status Register, Group x	$X508_H$	U, SV	U, SV P	27-42
GxQINR0	Queue 0 Input Register, Group x	$X510_H$	U, SV	U, SV P	27-44
GxQ0R0	Queue 0 Register 0, Group x	$X50C_H$	U, SV	U, SV P	27-46
GxQBUR0	Queue 0 Backup Register, Group x	$X510_H$	U, SV	U, SV P	27-48
GxQCTRL3	Queue 3 Source Control Register, Group x	$X540_H$	U, SV	U, SV P	27-50
GxQMR3	Queue 3 Mode Register, Group x	$X544_H$	U, SV	U, SV P	27-52
GxQSR3	Queue 3 Status Register, Group x	$X548_H$	U, SV	U, SV P	27-54
GxQINR3	Queue 3 Input Register, Group x	$X550_H$	U, SV	U, SV P	27-56
GxQ0R3	Queue 3 Register 0, Group x	$X54C_H$	U, SV	U, SV P	27-58

Table 27-11 Registers Overview (cont'd)

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Page Num.
			Read	Write	
GxQBUR3	Queue 3 Backup Register, Group x	$X550_H$	U, SV	U, SV P	27-60
GxTRCTR	Trigger Control Register, Group x	$X550_H$	U, SV	U, SV P	27-62
GxASCTRL	Autoscan Source Control Register, Group x	$X520_H$	U, SV	U, SV P	27-68
GxASMR	Autoscan Source Mode Register, Group x	$X524_H$	U, SV	U, SV P	27-70
GxASSEL	Autoscan Source Channel Select Register, Group x	$X528_H$	U, SV	U, SV P	27-72
GxASPND	Autoscan Source Pending Register, Group x	$X52C_H$	U, SV	U, SV P	27-72
BRSCTRL	Background Request Source Control Register	0200_H	U, SV	U, SV P	27-74
BRSMR	Background Request Source Mode Register	0204_H	U, SV	U, SV P	27-76
BRSSELx	Background Request Source Channel Select Register, Group x	$018Y_H$	U, SV	U, SV P	27-78
BRSPNDx	Background Request Source Channel Pending Register, Group x	$01CY_H$	U, SV	U, SV P	27-79
GxCHCTRy	Channel x Control Register	$X60Y_H$	U, SV	U, SV P	27-91
GxICLASS0	Input Class Register 0, Group x	$X4A0_H$	U, SV	U, SV P	27-94
GxICLASS1	Input Class Register 1, Group x	$X4A4_H$	U, SV	U, SV P	27-94
GLOBICLASS0	Input Class Register 0, Global	$00A0_H$	U, SV	U, SV P	27-94
GLOBICLASS1	Input Class Register 1, Global	$00A4_H$	U, SV	U, SV P	27-94
GxALIAS	Alias Register	$X4B0_H$	U, SV	U, SV P	27-97

Table 27-11 Registers Overview (cont'd)

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Page Num.
			Read	Write	
GxBOUND	Boundary Select Register, Group x	$X4B8_H$	U, SV	U, SV P	27-99
GLOBBOUND	Global Boundary Select Register	$00B8_H$	U, SV	U, SV P	27-99
GxBFL	Boundary Flag Register, Group x	$X4C8_H$	U, SV	U, SV P	27-105
GxBFLS	Boundary Flag Software Register, Group x	$X4CC_H$	U, SV	U, SV P	27-106
GxBFLC	Boundary Flag Control Register, Group x	$X4D0_H$	U, SV	U, SV P	27-107
GxBFLNP	Boundary Flag Node Pointer Register, Group x	$X4D4_H$	U, SV	U, SV P	27-108
GxRCRy	Group x Result Control Register y	$X68Y_H$	U, SV	U, SV P	27-112
GxRESy	Group x Result Register y	$X70Y_H$	U, SV	U, SV P	27-114
GxRESDy	Group x Result Register y (debug view)	$X78Y_H$	U, SV	U, SV P	27-116
GLOBRCR	Global Result Control Register	0280_H	U, SV	U, SV P	27-117
GLOBRES	Global Result Register	0300_H	U, SV	U, SV P	27-118
GLOBRESD	Global Result Register (debug view)	0380_H	U, SV	U, SV P	27-118
GxVFR	Valid Flag Register, Group x	$X5F8_H$	U, SV	U, SV P	27-120
GxSYNCTR	Synchronization Control Register	$X4C0_H$	U, SV	U, SV P	27-135
GLOBTF	Global Test Function Register	0160_H	U, SV	U, SV P	27-142
GLOBTE	Global Test Enable Register	0164_H	U, SV	U, SV P	27-144

Table 27-11 Registers Overview (cont'd)

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Page Num.
			Read	Write	
GxEMUXCTR	External Multiplexer Control Register, Group x	$X5F0_H$	U, SV	U, SV P	27-150
EMUXSEL	External Multiplexer Select Register	$03F0_H$	U, SV	U, SV P	27-152
GxSEFLAG	Source Event Flag Register, Group x	$X588_H$	U, SV	U, SV P	27-154
GxCEFLAG	Channel Event Flag Register, Group x	$X580_H$	U, SV	U, SV P	
GxREFLAG	Result Event Flag Register, Group x	$X584_H$	U, SV	U, SV P	27-155
GxSEFCLR	Source Event Flag Clear Register, Group x	$X598_H$	U, SV	U, SV P	27-156
GxCEFCLR	Channel Event Flag Clear Register, Group x	$X590_H$	U, SV	U, SV P	
GxREFCLR	Result Event Flag Clear Register, Group x	$X594_H$	U, SV	U, SV P	27-157
GLOBEFLAG	Global Event Flag Register	$00E0_H$	U, SV	U, SV P	27-158
GxSEVNP	Source Event Node Pointer Register, Group x	$X5C0_H$	U, SV	U, SV P	27-159
GxCEVNP0	Channel Event Node Pointer Register 0, Group x	$X5A0_H$	U, SV	U, SV P	27-160
GxCEVNP1	Channel Event Node Pointer Register 1, Group x	$X5A4_H$	U, SV	U, SV P	27-161
GxREVNP0	Result Event Node Pointer Register 0, Group x	$X5B0_H$	U, SV	U, SV P	27-162
GxREVNP1	Result Event Node Pointer Register 1, Group x	$X5B4_H$	U, SV	U, SV P	27-163
GLOBEVNP	Global Event Node Pointer Register	0140_H	U, SV	U, SV P	27-164
GxSRACT	Service Request Software Activation Trigger, Group x	$X5C8_H$	U, SV	U, SV P	27-166

27.14.3 Analog Module Connections in the TC21x/TC22x/TC23x

The VADC module accepts a number of analog input signals. The analog input multiplexers select the input channels to be converted from the signals available in this product.

Note: If an analog input channel is connected to an I/O port pin, make sure the output driver and/or the digital input path are disabled.

The exact number of analog input channels and the available connection to port pins depend on the employed product type (see also **Table 27-8**). A summary of channels enclosing all versions of the TC21x/TC22x/TC23x can be found below.

Input channels marked “PDD” provide a pull-down device for pull-down diagnostics.

Input channels marked “MD” can activate the pullup and pulldown devices for multiplexer diagnostics.

Input channels marked “AltRef” can be selected as an alternate reference voltage for conversions on channels of the same group or for conversions on channels of the same cluster.

Input channels marked “noAltref” cannot select an alternate reference voltage, but only the corresponding standard reference voltage.

Input channels marked “(A)” can be selected via the alias feature (**Section 27.7.2**), not directly.

Table 27-12 Analog Connections in the TC21x/TC22x/TC23x

Signal	Dir.	Source/Destin.	Description
V_{AREF}	I	VAREF	positive analog reference
V_{AGND}	I	VAGND	negative analog reference
G0CH0 (AltRef)	I	AN0, P40.0	analog input channel 0 of group 0
G0CH1 (MD)	I	AN1, P40.1	analog input channel 1 of group 0, overlaid with G2CH0 (ADAS device)
G0CH2 (MD)	I	AN2, P40.2	analog input channel 2 of group 0
G0CH3	I	AN3, P40.3	analog input channel 3 of group 0
G0CH4 (noAltref)	I	AN4, P40.4	analog input channel 4 of group 0
G0CH5 (noAltref)	I	AN5, P40.5	analog input channel 5 of group 0
G0CH6 (noAltref)	I	AN6, P40.6	analog input channel 6 of group 0
G0CH7 (PDD, noAltref)	I	AN7, P40.7	analog input channel 7 of group 0
G0CH8	I	AN8, P40.8	analog input channel 8 of group 0
G0CH9 (MD)	I	AN9, P40.9	analog input channel 9 of group 0

Table 27-12 Analog Connections in the TC21x/TC22x/TC23x (cont'd)

Signal	Dir.	Source/Destin.	Description
G0CH10 (MD)	I	AN10, P40.10	analog input channel 10 of group 0
G0CH11	I	AN11, P40.11	analog input channel 11 of group 0
G0CH12 (A)	I	HP Bandgap (int.)	analog input channel 12 of group 0
G0CH13 (A)	I	VDDM/2 (int.)	analog input channel 13 of group 0
G1CH0 (AltRef)	I	AN12, P41.0	analog input channel 0 of group 1
G1CH1 (MD)	I	AN13, P41.1	analog input channel 1 of group 1, overlaid with G3CH0 (ADAS device)
G1CH2 (MD)	I	AN14, P41.2	analog input channel 2 of group 1
G1CH3 (PDD)	I	AN15, P41.3	analog input channel 3 of group 1
G1CH4	I	AN16, P41.4	analog input channel 4 of group 1
G1CH5	I	AN17, P41.5	analog input channel 5 of group 1
G1CH6	I	AN18, P41.6	analog input channel 6 of group 1
G1CH7	I	AN19, P41.7	analog input channel 7 of group 1
G1CH8	I	AN20, P41.8	analog input channel 8 of group 1
G1CH9 (MD)	I	AN21, P41.9	analog input channel 9 of group 1
G1CH10 (MD)	I	AN22, P41.10	analog input channel 10 of group 1
G1CH11	I	AN23, P41.11	analog input channel 11 of group 1
G1CH12 (A)	I	VDD (int.)	analog input channel 12 of group 1
G1CH13 (A)	I	VDD3/2 (int.)	analog input channel 13 of group 1
G2CH0	I	AN1, P40.1	analog input channel 0 of group 2, overlaid with G0CH1
G3CH0	I	AN13, P41.1	analog input channel 0 of group 3, overlaid with G1CH1

27.14.4 Digital Module Connections in the TC21x/TC22x/TC23x

The VADC module accepts a number of digital input signals and generates a number of output signals. This section summarizes the connection of these signals to other on-chip modules or to external resources via port pins.

Note: The control bitfields for triggers and gates select the corresponding multiplexer input. Values 0000_B … 1111_B select inputs with suffix -A … -P.

Table 27-13 Digital Connections in the TC21x/TC22x/TC23x

Signal	Dir.	Source/Destin.	Description
Gate Inputs for Each Group			
G0REQGTA	I	GTM_adc0_trig0	GTM ADC trigger 0
G1REQGTA	I	GTM_adc1_trig0	GTM ADC trigger 0
G2REQGTA	I	GTM_adc3_trig0	GTM ADC trigger 0
G3REQGTA	I	GTM_adc3_trig0	GTM ADC trigger 0
G0REQGTB	I	GTM_adc0_trig1	GTM ADC trigger 1
G1REQGTB	I	GTM_adc1_trig1	GTM ADC trigger 1
G2REQGTB	I	GTM_adc3_trig1	GTM ADC trigger 1
G3REQGTB	I	GTM_adc3_trig1	GTM ADC trigger 1
GxREQGTC	I	CCU6061 TRIG0	CCU6061 trigger output 0
GxREQGTD	I	CCU6061 TRIG1	CCU6061 trigger output 1
GxREQGTE	I	CCU6061 TRIG2	CCU6061 trigger output 2
GxREQGTF	I	-	Gating input F, group x
GxREQGTG	I	-	Gating input G, group x
GxREQGTH	I	-	Gating input H, group x
GxREQGTI	I (s)	-	Gating input I, group x
GxREQGTJ	I (s)	-	Gating input J, group x
GxREQGTK	I (s)	-	Gating input K, group x
GxREQGTL	I (s)	-	Gating input L, group x
GxREQGTM	I	eru_pdout_x	ERU pattern detection output x
GxREQGTN	I	-	Gating input N, group x
GxREQGTO	I	-	Gating input O, group x
GxREQGTP	I	[internal]	Extend inputs to selected internal trigger source (see **GxTRCTR (x = 0 - 3)**)

Table 27-13 Digital Connections in the TC21x/TC22x/TC23x (cont'd)

Signal	Dir.	Source/Destin.	Description
GxREQGTzSEL	O	GxREQTRzP[1)]	Selected gating signal of the respective source (z = 0 - 1)
Gate Inputs for Global Background Source			
BGREQGTA	I	GTM_adc2_trig0	GTM ADC trigger 0
BGREQGTB	I	GTM_adc2_trig1	GTM ADC trigger 1
BGREQGTC	I	CCU6061 TRIG0	CCU6061 trigger output 0
BGREQGTD	I	CCU6061 TRIG1	CCU6061 trigger output 1
BGREQGTE	I	CCU6061 TRIG2	CCU6061 trigger output 2
BGREQGTF	I	-	Gating input F, background source
BGREQGTG	I	-	Gating input G, background source
BGREQGTH	I	-	Gating input H, background source
BGREQGTI	I (s)	-	Gating input I, background source
BGREQGTJ	I (s)	-	Gating input J, background source
BGREQGTK	I (s)	-	Gating input K, background source
BGREQGTL	I (s)	-	Gating input L, background source
BGREQGTM	I	-	Gating input M, background source
BGREQGTN	I	-	Gating input N, background source
BGREQGTO	I	-	Gating input O, background source
BGREQGTP	I	-	Gating input E, background source
BGREQGTSEL	O	BGREQTRP[1)]	Selected gating signal
Trigger Inputs for Each Group			
GxREQTRA	I	CCU60_SR3	CCU60 service request output 3
GxREQTRB	I	CCU61_SR3	CCU61 service request output 3
GxREQTRC	I	-	Trigger input C, group x
GxREQTRD	I	-	Trigger input D, group x
GxREQTRE	I	-	Trigger input E, group x
GxREQTRF	I	-	Trigger input F, group x
GxREQTRG	I	-	Trigger input G, group x
GxREQTRH	I	eru_iout_x	ERU interrupt output x
GxREQTRI	I (s)	-	Trigger input I, group x
GxREQTRJ	I (s)	-	Trigger input J, group x

Table 27-13 Digital Connections in the TC21x/TC22x/TC23x (cont'd)

Signal	Dir.	Source/Destin.	Description
GxREQTRK	I (s)	-	Trigger input K, group x
GxREQTRL	I (s)	-	Trigger input L, group x
GxREQTRM	I	vadc_gxsr1	Service request 1, group x
GxREQTRN	I	vadc_c0sr1	Service request 1, common group 0
GxREQTRO	I	vadc_c1sr1	Service request 1, common group 1
GxREQTRzP	I	GxREQGTzSEL[1)]	Extend triggers to selected gating input of the respective source (z = 0 - 1)
GxREQTRzSEL	O	-	Selected trigger signal of the respective source (z = 0 - 1)
Trigger Inputs for Global Background Source			
BGREQTRA	I	CCU60_SR3	CCU60 service request output 3
BGREQTRB	I	CCU61_SR3	CCU61 service request output 3
BGREQTRC	I	-	Trigger input C, background source
BGREQTRD	I	-	Trigger input D, background source
BGREQTRE	I	-	Trigger input E, background source
BGREQTRF	I	-	Trigger input F, background source
BGREQTRG	I	eru_iout_3	ERU interrupt output 3
BGREQTRH	I	eru_iout_4	ERU interrupt output 4
BGREQTRI	I (s)	eru_iout_6	ERU interrupt output 6
BGREQTRJ	I (s)	eru_iout_7	ERU interrupt output 7
BGREQTRK	I (s)	-	Trigger input K, background source
BGREQTRL	I (s)	-	Trigger input L, background source
BGREQTRM	I	-	Trigger input M, background source
BGREQTRN	I	vadc_c0sr1	Service request 1, common group 0
BGREQTRO	I	vadc_c1sr1	Service request 1, common group 1
BGREQTRP	I	BGREQGTSEL[1)]	Extend triggers to selected gating input of the background source
BGREQTRSEL	O	-	Selected trigger signal of the background source
System-Internal Connections			

Table 27-13 Digital Connections in the TC21x/TC22x/TC23x (cont'd)

Signal	Dir.	Source/Destin.	Description
otgb0[15:0]	O	OTGM	Alternate trigger buses for additional trace signals indicating the input signal sample phase (see)
otgb1[15:0]	O	OTGM	
G0ARBCNT	O	-	Outputs a (count) pulse for each arbiter round
G1ARBCNT	O	-	
G2ARBCNT	O	-	
G3ARBCNT	O	-	
GxSR0	O	ICU	Service request 0 of group x
GxSR1	O	ICU	Service request 1of group x
GxSR2	O	ICU	Service request 2 of group x
GxSR3	O	ICU	Service request 3 of group x
C0SR0	O	ICU GTM_TIM0_CH0	Service request 0 of common block 0
C0SR1	O	ICU GTM_TIM0_CH2	Service request 1 of common block 0
C0SR2	O	ICU GTM_TIM0_CH4	Service request 2 of common block 0
C0SR3	O	ICU GTM_TIM0_CH6	Service request 3 of common block 0
C1SR0	O	GTM_TIM0_CH1	Service request 0 of common block 1
C1SR1	O	GTM_TIM0_CH3	Service request 1 of common block 1
C1SR2	O	GTM_TIM0_CH5	Service request 2 of common block 1
C1SR3	O	GTM_TIM0_CH7	Service request 3 of common block 1
EMUX00	O	P02.6, P33.3	Control of external analog multiplexer interface 0
EMUX01	O	P02.7, P33.2	
EMUX02	O	P02.8, P33.1	
EMUX10	O	P00.6, P33.6	Control of external analog multiplexer interface 1
EMUX11	O	P00.7, P33.5	
EMUX12	O	P00.8, P33.4	
CBFLOUT0	O	-	Common boundary flag output 0
CBFLOUT1	O	-	Common boundary flag output 1
CBFLOUT2	O	-	Common boundary flag output 2

Table 27-13 Digital Connections in the TC21x/TC22x/TC23x (cont'd)

Signal	Dir.	Source/Destin.	Description
CBFLOUT3	O	-	Common boundary flag output 3
VADCG0BFL0	O	P33.4	Boundary flag 0 output of group 0
G0BFL0	O	GTM_ stat_in00_[0]	Boundary flag 0 level of group 0
G0BFSEL0	I	0	Boundary flag 0 (group 0) source select
G0BFDAT0	I	0	Boundary flag 0 (group 0) alternate data
VADCG0BFL1	O	P33.5	Boundary flag 1 output of group 0
G0BFL1	O	GTM_ stat_in00_[1]	Boundary flag 1 level of group 0
G0BFSEL1	I	0	Boundary flag 1 (group 0) source select
G0BFDAT1	I	0	Boundary flag 1 (group 0) alternate data
VADCG0BFL2	O	P33.6	Boundary flag 2 output of group 0
G0BFL2	O	-	Boundary flag 2 level of group 0
G0BFSEL2	I	0	Boundary flag 2 (group 0) source select
G0BFDAT2	I	0	Boundary flag 2 (group 0) alternate data
VADCG0BFL3	O	P33.7 CCU60_CTRAPC	Boundary flag 3 output of group 0
G0BFL3	O	-	Boundary flag 3 level of group 0
G0BFSEL3	I	0	Boundary flag 3 (group 0) source select
G0BFDAT3	I	0	Boundary flag 3 (group 0) alternate data
VADCG1BFL0	O	P33.0 P00.4	Boundary flag 0 output of group 1
G1BFL0	O	GTM_ stat_in01_[0]	Boundary flag 0 level of group 1
G1BFSEL0	I	-	Boundary flag 0 (group 1) source select
G1BFDAT0	I	-	Boundary flag 0 (group 1) alternate data
VADCG1BFL1	O	P33.1 P00.5	Boundary flag 1 output of group 1
G1BFL1	O	GTM_ stat_in01_[1]	Boundary flag 1 level of group 1
G1BFSEL1	I	-	Boundary flag 1 (group 1) source select
G1BFDAT1	I	-	Boundary flag 1 (group 1) alternate data

Table 27-13 Digital Connections in the TC21x/TC22x/TC23x (cont'd)

Signal	Dir.	Source/Destin.	Description
VADCG1BFL2	O	P33.2 P00.6	Boundary flag 2 output of group 1
G1BFL2	O	-	Boundary flag 2 level of group 1
G1BFSEL2	I	0	Boundary flag 2 (group 1) source select
G1BFDAT2	I	0	Boundary flag 2 (group 1) alternate data
VADCG1BFL3	O	P33.3 P00.7	Boundary flag 3 output of group 1
G1BFL3	O	-	Boundary flag 3 level of group 1
G1BFSEL3	I	0	Boundary flag 3 (group 1) source select
G1BFDAT3	I	0	Boundary flag 3 (group 1) alternate data
VADCG2BFLx	O	-	Boundary flag x output of group 2
G2BFLx	O	-	Boundary flag x level of group 2
G2BFSELx	I	0	Boundary flag x (group 2) source select
G2BFDATx	I	0	Boundary flag x (group 2) alternate data
VADCG3BFLx	O	-	Boundary flag x output of group 3
G3BFLx	O	-	Boundary flag x level of group 3
G3BFSELx	I	0	Boundary flag x (group 3) source select
G3BFDATx	I	0	Boundary flag x (group 3) alternate data

1) Internal signal connection.

27.15 Use Case Example for VADC

This section provides a code example for the VADC module to give an overview about the core functionality. The code example realizes an event triggered analog to digital conversion. The result is stored as a 12-bit digital value and triggers a result service request.

The VADC contains several ADC groups (G0 … G3). Each group is connected to up to 8 analog input channels via a multiplexer. The following code example initializes one of the groups and uses one static allocated analog channel as input.

The code example realizes an analog-to-digital conversion by using the background source. The conversion can either run continuously in autoscan mode or can be triggered by an event, like in the following example. This is useful if the conversion needs to be executed at certain times. The result from the most recent conversion is acceccible in the global result register **GLOBRES**. A result service request is activated after each conversion. The initialization is done in the following order:

1. Load the global VADC module registers
2. Enable the analog/digital converter 0 (G0)
3. Select channel 0 of G0 as analog input
4. Initialize the conversion result service request
5. Start first conversion

Step Description to Initialize the VADC Module:

(Line 1) reset ENDINIT to get access to ENDINIT protected registers. (see also ENDINIT protection in chapter SCU).

(Line 2) enable control of the module, in clock control register **CLC**.

(Line 3) read back (dummy variable has to be defined). The reading process ensures that the write process from line 2 is done.

(Line 4) set ENDINIT to lock the protected register again.

(Line 5) set group 0 to normal operation mode. This activates the converter of group 0. (see also **Section 27.4.1** and **GxARBCFG (x = 0 - 3)**).

(Line 6) load the global configuration register GLOBCFG with the divider factor for the analog internal clock (see also **Figure 27-7**). Set DIVWC[15] to enable write access and initiate the start-up calibration by setting SUCAL[31]. The calibration is necessary to get a precise conversion.

(Line 7) This line enables the arbitration slot 2, that's the background scan source slot which the examples uses. (see also Arbitration Priority Register **GxARBPR (x = 0 - 3)**)

(Line 8) This line sets RESTBS[20], so each conversion result from group 0 input channel 0 is now stored in the global result register GLOBRES. (see also GxCHCTRy)

(Line 9) This line selects input channel 0 of group 0 as part of the background scan sequence. (see also **BRSSELx (x = 0 - 3)**).

(Line 10) By setting ENGT[0..1]=01 a conversion request can be issued for every channel which was enabled in line 10. here only channel 0. (see also **BRSMR**)

(Line 11) This line enables the group 0 service request in the service request control register SRC_VADCG0SR0 and sets the interrupt priority to VADC0INT (1...255)

(Line 12) The interruptHandlerInstall function gets called (this function has to be written first, see interrupt handler example in chapter IR for more details). This function installs the interrupt service routine (ISR) entry address in the interrupt vector array with the priority VADC0INT.

(Line 13) This command enables the result interrupt. It occurs after a result event.

(Line 14) wait until start-up calibration (start in line 6) is done.

(Line 15) setting LDEV=1 generates a load event which starts a conversion.

Note: The conversion result ISR function prototype would be: void VADC_SCAN_irq (void); here could be your ISR code;

Basic Initialization of the VADC:

```
//=========================> Load global module registers
(1)  SCU_vResetENDINIT (0);     // Access to ENDINIT-prot. reg.
(2)  VADC_CLC = 0x0000;         // Enable module clock and ctrl.
(3)  dummy = VADC_CLC;          // Read back ensures write oper.
(4)  SCU_vSetENDINIT (0);       // Lock ENDINIT-protected reg.
//=========================> Enable converter for group 0
(5)  VADC_G0ARBCFG = 0x3;       // ANONC = 11, analog converter ON
(6)  VADC_GLOBCFG = (1<<31)     /* SUCAL = 1, start-up calib. */ \
                    |(1<<15)    /* DIVWC = 1, enable write */ \
                    | 0x9;      // DIVA = 9 (clock prescaler)
(7)  VADC_G0ARBPR = (1<<26);    // AREN2 = 1, enable arb. slot 2
                                //          (= background source)
(8)  VADC_G0CHCTR0 = (1<<20);   // RESTBS = 1, global result reg.
(9)  VADC_BRSSEL0 = 0x1;        // Select CH0 of group 0 for scan
(10) VADC_BRSMR = 0x1;          // ENGT = 10B, enable conv. req.
//=========================> Init and install service request
(11) SRC_VADCCG0SR0 = (1<<10)   /* Enable SR node 0, group 0 */ \
                     |VADC0INT; // Set prio to <VADC0INT>(1..255)
(12) interruptHandlerInstall (VADC0INT, & VADC_SCAN_irq);
(13) VADC_GLOBRCR = (1<<31);    // SRGEN = 1, result service requ.
//=========================> Wait for completion of startup cal.
(14) while((VADC_G0ARBCFG.U & 0x30000000) != 0x20000000);
                                // CALS = 1, CAL = 0: calibr. done
//=========================> Start a conversion
(15) VADC_BRSMR |= (1<<9);      // LDEV = 1, generate a load event
```

Note: Before a service request can occur, the service request system must be globally enabled. The Interrupt Control Register (ICR) holds the global interrupt enable bit (ICR.IE) which enables the CPU service request system. Most compilers support the attribute (or similar) to set this bit (see Architecture Manual for more details): ***__enable();***

28 Input Output Monitor (IOM)

28.1 Overview

The Input Output Monitor (IOM) module serves as a smart I/O comparison unit, observing and checking for the correct operation of system peripheral outputs that may serve and/or control externally-attached hardware, as well as the correct operation of the hardware itself, including any sensors whose signals may serve as an input to the monitoring function.

The monitoring function may be achieved in a number of configurable ways depending upon the application and the type of safety measure to be deployed. (Examples of such measures are included later-on within this chapter). Triggering via software is also permissible in this manner.

Monitor and reference signals (where needed) are taken from applicable system peripherals (i.e. Capture/Compare Units and General Purpose Timer Units) where they connect to internal padlogic (that feeds/controls I/O Ports), which may or may not be driven to the pads (configuration dependent). In this manner connectivity can be made to a signal that is to be driven externally, and to another signal that may serve as a reference, although may not be driven off-chip. A driven signal may also serve as a reference in order to compare against another that may be provided via a GPIO input, e.g. from a sensor external to the device.

Up to 16 monitoring points can be configured, with the capability to generate a system event that may be driven from one individual, several individual or multiple occurrences of a monitored condition, or a combination of several depending upon configuration.

28.2 Features

- 16 Filter & Prescaler Channels, each connected to a specific pad (Pn_IN), the output of which to serve as a monitor or reference signal.
- 16 Logic Analyzer Modules (LAMs), with each channel comprising a mux to independently select monitor and reference signal inputs, with which to undertake the compare. A local event (i.e. internal to IOM) is generated when parity or disparity occurs between the selected monitor and reference signals. Configurable via register settings.
- 1 EXOR combiner, configurable (via register) to select a range of upto 8 GTM inputs in order to generate a combined signal to serve as a reference.
- Event Combiner Module (ECM), that takes the 16 local event signals (1 from each of the LAM modules) and generates a system event signal (connecting to system Safety Management Unit) from a single, combination or multiple local event(s). Configurable via register settings.

- System Peripheral Bus (SPB) interface for configuration and status register interaction.

28.3 Interfaces

The IOM features the following interfaces:

- System Peripheral Bus (SPB) - for the configuration of constituent parts of the IOM and the interrogation of associated status registers/bitfields.
- Multiple port logic/module connectivity, serving as reference or monitor signals. See SoC Integration section within this chapter for device-specific connections to GTM, GPT12, QSPI, CCU, GPIO (for GTM, etc.).
- Safety Management Unit (SMU) - a single global event signal used inform the SMU of a generated event within the IOM.
- System Signals - FPI Clock, GTM Clock, System Reset.

Note: The functionality of the IOM is independent from system debug, and no connectivity or mode of operation exists for this purpose.

28.4 Kernel Description

In addition to the Logic Analyzer Modules, Filter & Prescaler Blocks and Event Combiner Module, the IOM also features a System Peripheral Bus (SPB) interface for configuration and status register access. Note: All the configuration registers are protected by the register access protection mechanism (global definition).

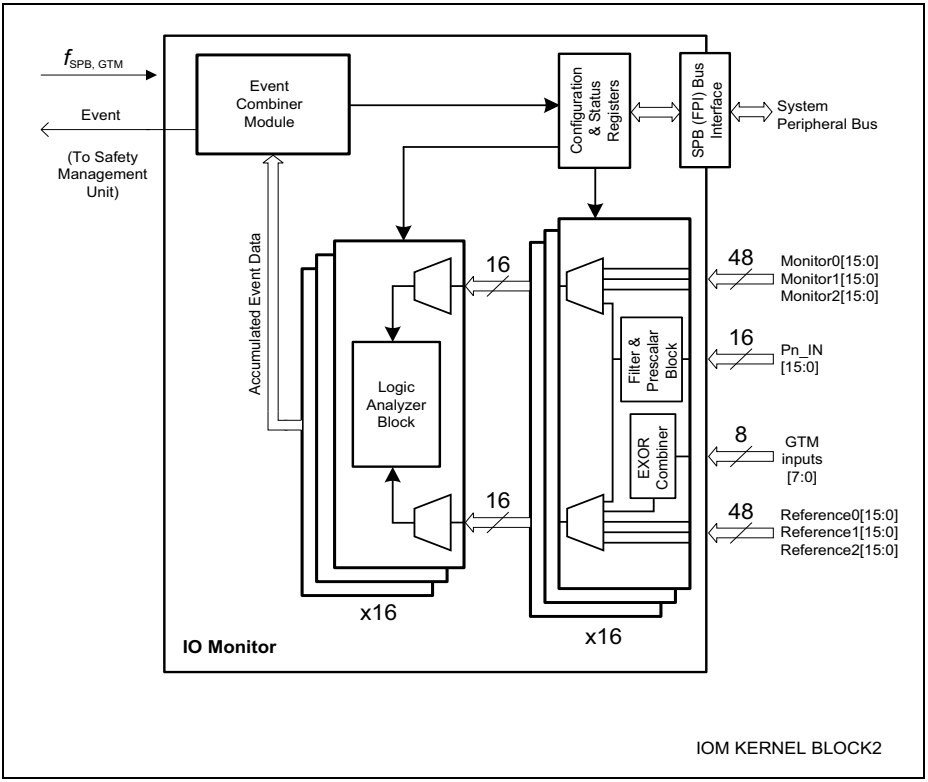


Figure 28-1 IO Monitor Block Diagram

The operational clock for the IOM is derived by combining the SPB and GTM clocks as a logical OR, the resultant frequency of which is therefore determined to be the higher of the two frequencies (assuming a synchronous relationship between the two, both being derived from the same system-level clock). This provides the capability for the IOM to sample GTM-related signals when the SPB (FPI) clock is lower than that of the GTM within the system.

28.5 Filter & Prescaler Channel Description

The IOM instances 16 Filter & Prescaler Channels(FPC's) for the purposes of preconditioning and filtering the signals received from the pads (Pn_IN inputs) that may be used as a reference or monitor.

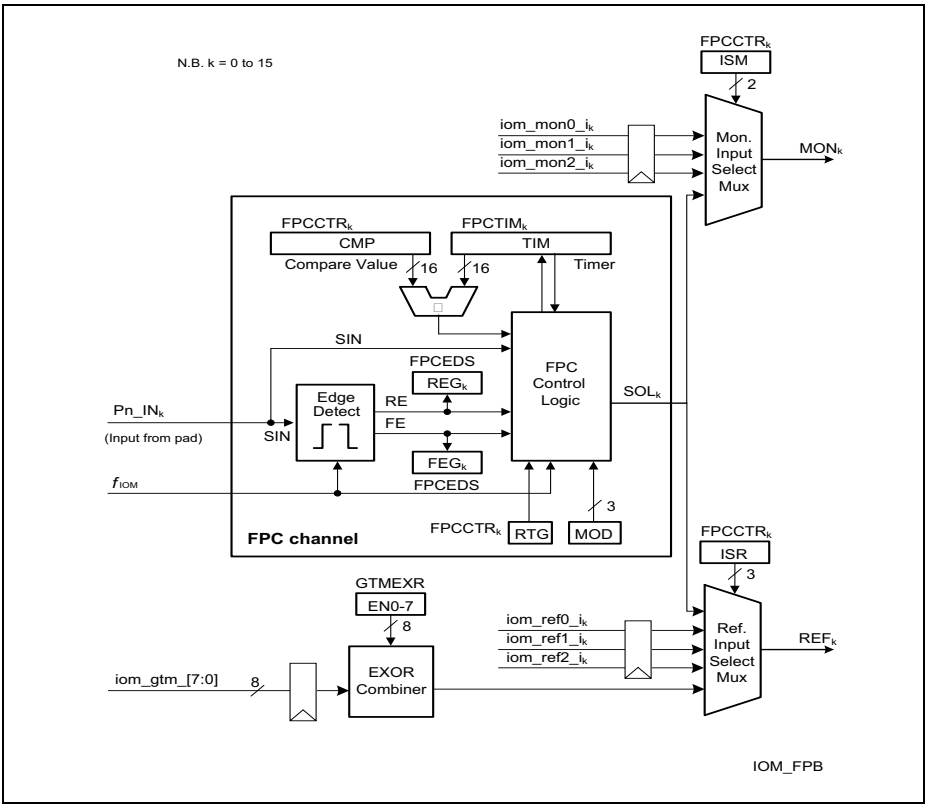


Figure 28-2 Monitor & Reference Selection logic incorporating Filter and Prescaler channel

As shown in **Figure 28-2**, each FPC channel (cell) is equipped with an signal input multiplexer, a clock multiplexer, an edge detection circuitry, a 16-bit timer, a 16-bit compare register, a 16-bit comparator, and a FPC control circuitry. The edge detection circuitry detects respective edges for the prescaler modes and detects glitches in all other modes.

FPC Registers

The following registers are assigned to the Filter and Prescaler Cells FPC_k (k = 0-15):

- FPCESR = Filter and Prescaler Cells Rising and Falling Edge Status Register.
- $FPCCTR_k$ = Filter and Prescaler Cell Control Register k
- $FPCTIM_k$ = Filter and Prescaler Cell Timer Register k

FPC Operating Modes

Each filter and prescaler cell can be individually configured to operate in one of the following operating modes:

- Delayed Debounce Filter Mode on both edges
- Immediate Debounce Filter Mode on both edges
- Rising edge: Immediate Debounce Filter Mode, falling edge: No filtering
- Rising edge: No filtering, falling edge: Immediate Debounce Filter Mode
- Rising edge: Delayed Debounce Filter Mode, falling edge: Immediate Debounce Filter Mode
- Rising edge: Immediate Debounce Filter Mode, falling edge: Delayed Debounce Filter Mode
- Prescaler Mode (triggered by edge detection circuitry on rising edge)
- Prescaler Mode (triggered by edge detection circuitry on falling edge)

The operation mode is selected by bit field $FPCCTR_k$.MOD.

FPC Input Signal notes

The maximum FPC input signal frequency must be less than or equal to the sampling rate ($f_{FPI}/2$).

FPC Output Signal notes

A level output, representative signal SOL_k, indicating the direction of the detected signal transition, that can be configured to connect to the Logic Analyzer $Module_m$.

Delayed Debounce Filter Mode

In Delayed Debounce Filter Mode, the signal input SIN is filtered from all signal transitions and glitches with a width smaller than the selected clock period length multiplied by the compare register value.

The input signal SIN is analyzed at the selected filter clock rate of f_{IOM}. If the state of the input sample differs from the current output signal value, the 16-bit timer is increased by one. When the timer register $FPCTIM_k$ is not in its idle state (0000_H) **and** the state of the input sample matches the current output signal value, the 16-bit timer is decremented by one (see **Figure 28-3**); if bit $FPCCTR_k$.RTG is set, the timer will be set to idle state again (see **Figure 28-4**). A rising or falling edge, occurring on the signal input line SIN when the timer is greater than zero but less than the compare value, sets the corresponding glitch flag FPCRES.REG (on rising edge glitch) or FPCFES.FEG (on falling edge glitch). When the timer matches the 16-bit compare value stored in $FPCCTR_k$.CMP (timer threshold), the level output signal line SOL_k is inverted, and the timer is reset to 0000_H. The rising/falling edge glitch flags must be reset by software.

The filter is by-passed if the compare value $FPCCTR_k$.CMP is programmed to zero (0000_H). In this case, the input signal is directly copied to the output signal.

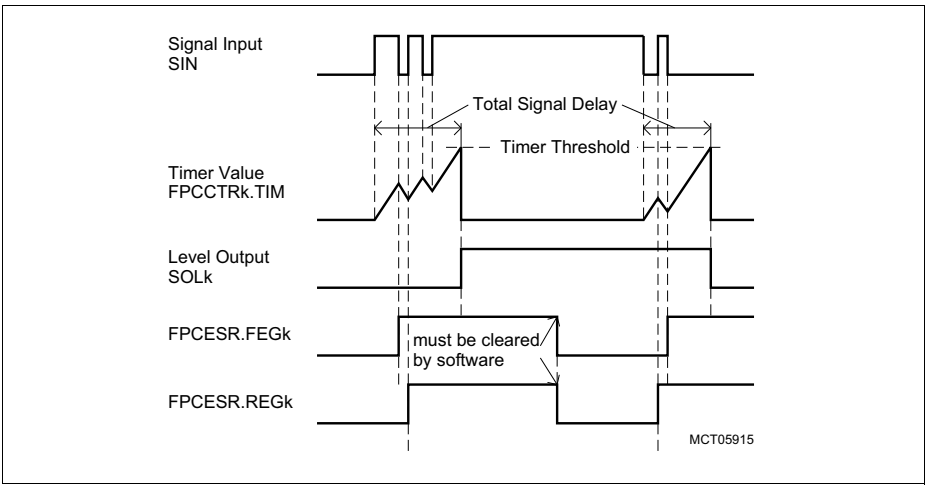


Figure 28-3 FPC Delayed Debounce Filter Algorithm with Timer Decrement

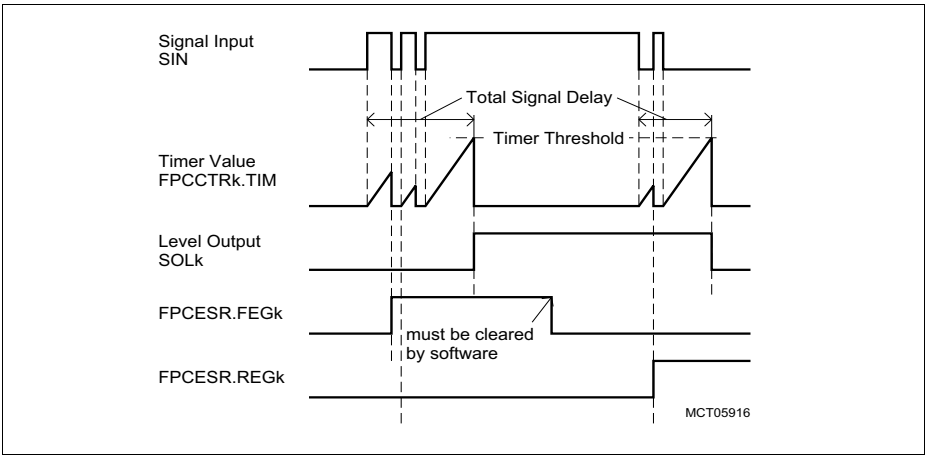


Figure 28-4 FPC Delayed Debounce Filter Algorithm with Timer Reset

The total signal delay from input to output depends on the programmed compare register value, the number of high-frequency pulses (glitches) during the filter operating time, and the timer behaviour in case of a glitch (decrement or reset).

The FPC Delayed Debounce Filter Mode is selected by:

- $FPCCTR_k.MOD = 000_B$

Immediate Debounce Filter Mode

In Immediate Debounce Filter Mode, the input signal is filtered from signal transitions and glitches arriving a programmable time after an input signal edge detection (see **Figure 28-5**).

The input signal SIN is sampled with f_{FPI} and the input signal SIN edge detection is also performed with f_{FPI}. The further analysis (e.g. filter timer increment, glitch detection) is also achieved with f_{FPI}.

As long as the timer is reset, the FPC control circuitry copies the sampled input value directly to the level output signal line SOLk. When a rising or falling edge occurs on the signal input line SIN and the 16-bit compare value FPCCTR$_k$.CMP is not zero, the timer is enabled to be increased by the selected clock and the copy mechanism is disabled. When the timer value FPCTIM$_k$.TIM matches the compare value FPCCTR$_k$.CMP, the timer is reset and the copy mechanism is enabled again. A rising or falling edge, occurring on SIN while the timer is greater than zero but less than the compare value, sets the corresponding glitch flag FPCRES.REG (on rising edge glitch) or FPCFES.FEG (on falling edge glitch). The rising/falling edge glitch flags must be reset by software.

The filter is by-passed if the compare value FPCCTR$_k$.CMP is programmed to zero (0000_H). In this case, the input signal is directly copied to the output signal without any disable periods.

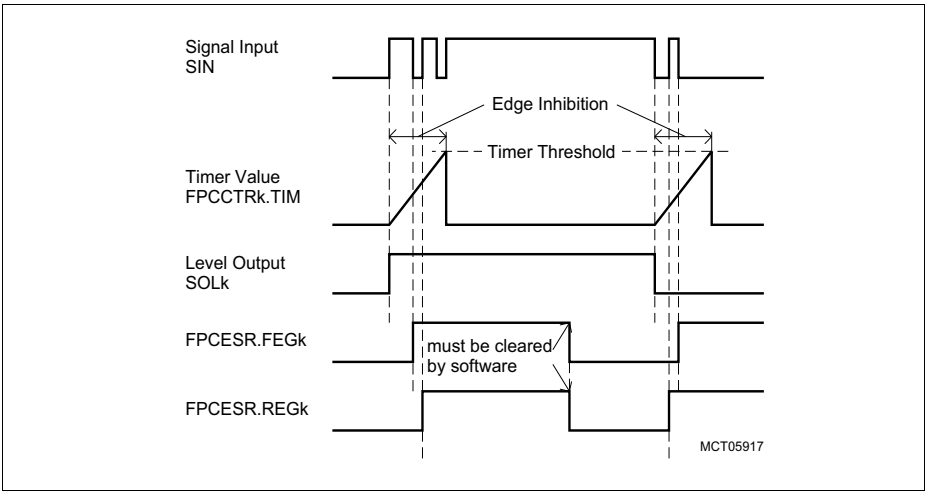


Figure 28-5 FPC Immediate Debounce Filter Algorithm on Both Edges

Note: During the last clock cycle of edge inhibition time (where timer value is equal to the compare value) an input signal glitch will be filtered but the corresponding glitch status flag in register FPCRES/FPCFES is not set.

The Immediate Debounce Filter can be enabled only for one edge, either rising or falling. In this case, the signal output follows the signal input value immediately after the timer threshold of the filtered edge is reached, without re-starting the timer (see **Figure 28-6**).

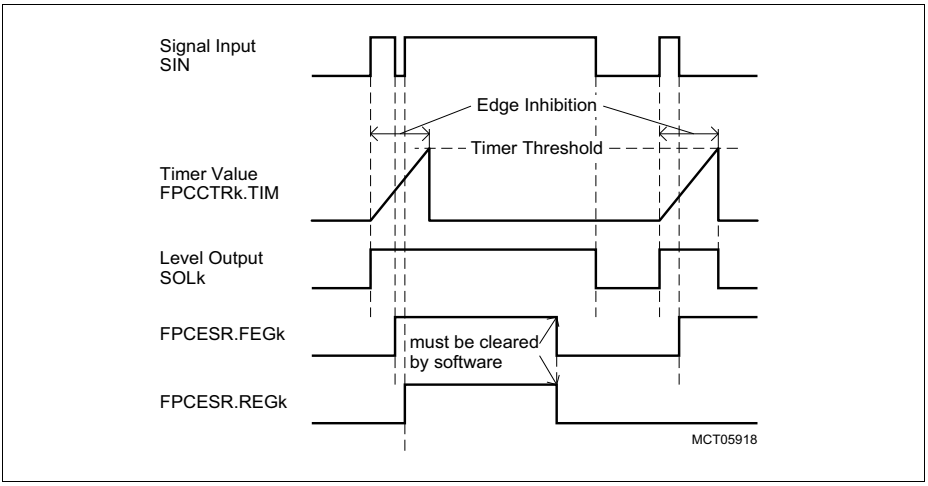


Figure 28-6 FPC Immediate Debounce Filter Algorithm on Rising Edge only

The FPC Immediate Debounce Filter Modes are selected by:

- $FPCCTR_k.MOD = 001_B$: Immediate Debounce Filter Mode on both edges
- $FPCCTR_k.MOD = 010_B$: Immediate Debounce Filter Mode on rising edge only, no filtering on falling edge.
- $FPCCTR_k.MOD = 011_B$: Immediate Debounce Filter Mode on falling edge only, no filtering on rising edge.

Mixed Filter Modes

In the Mixed Filter Modes, one edge of a signal is filtered in the Delayed Debounce Mode, and the other edge is filtered in the Immediate Debounce Mode. The Debounce Mode is switched when the timer threshold is reached. Note that both filter modes use the same timer threshold in this case (see **Figure 28-7**, demonstrating Delayed Debounce Mode with Timer Decrement on Rising Edge and Immediate Debounce of on Falling Edge).

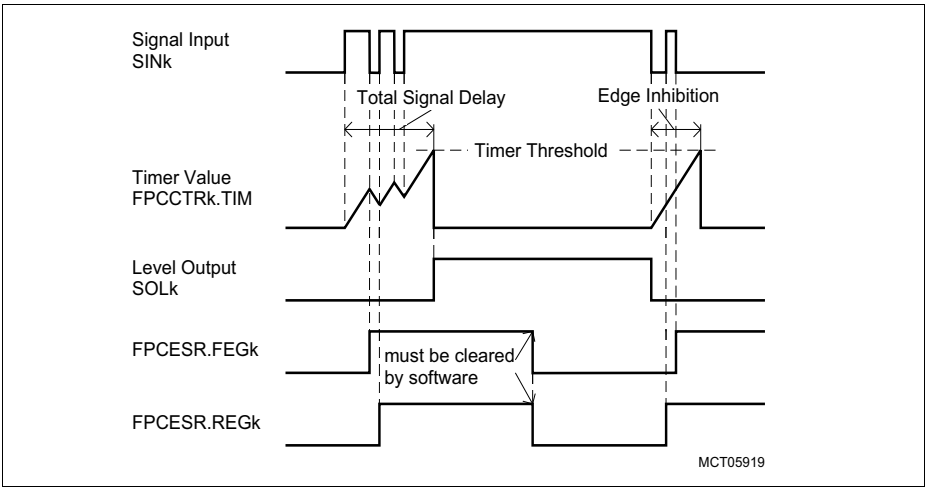


Figure 28-7 FPC Mixed Filter Algorithm

The FPC Mixed Filter Modes are selected by:

- $FPCCTR_k.MOD = 100_B$: Delayed Debounce Filter Mode on rising edge
  Immediate Debounce Filter Mode on falling edge
- $FPCCTR_k.MOD = 101_B$: Immediate Debounce Filter Mode on rising edge
  Delayed Debounce Filter Mode on falling edge

Prescaler Mode

In Prescaler Mode, the input signal is sampled and analyzed with f_{FPI}. The FPC control circuitry counts each rising (or falling) edge of the input signal. When the timer value matches the compare value:

- one IOM module clock pulse is generated at the level output signal SOL_k
- the timer $FPCTIM_k$.TIM is reset to 0000_H

Figure 28-8 shows a divide-by-6 operation using the FPC in Prescaler Mode with trigger on rising edge selected.

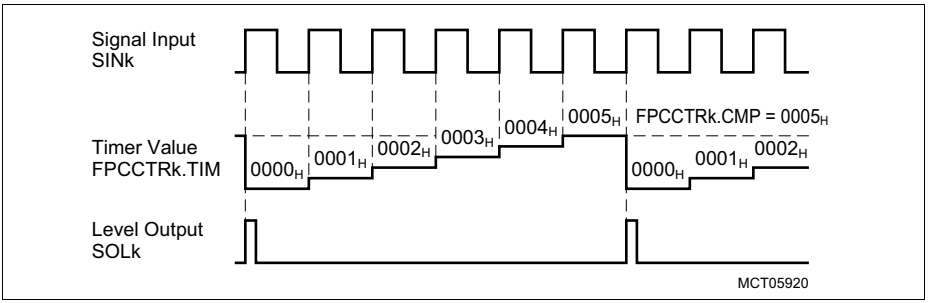


Figure 28-8 FPC Prescaler Mode

For a divide-by-n operation, the compare value $FPCCTR_k$.CMP must be set to n - 1.

The FPC Prescaler Modes are selected by:

- $FPCCTR_k$.MOD = 110_B: Prescaler Mode triggered by edge detection circuitry on rising edge
- $FPCCTR_k$.MOD = 111_B: Prescaler Mode triggered by edge detection circuitry on falling edge

28.6 EXOR Combiner Description

The EXOR Combiner can be configured to accepts upto 8 external inputs from GTMmodule outputs, used for replicate a typical *combined* signal from an automotive multi-phase motor. The output from which can then be used as a reference to monitor such a motor.

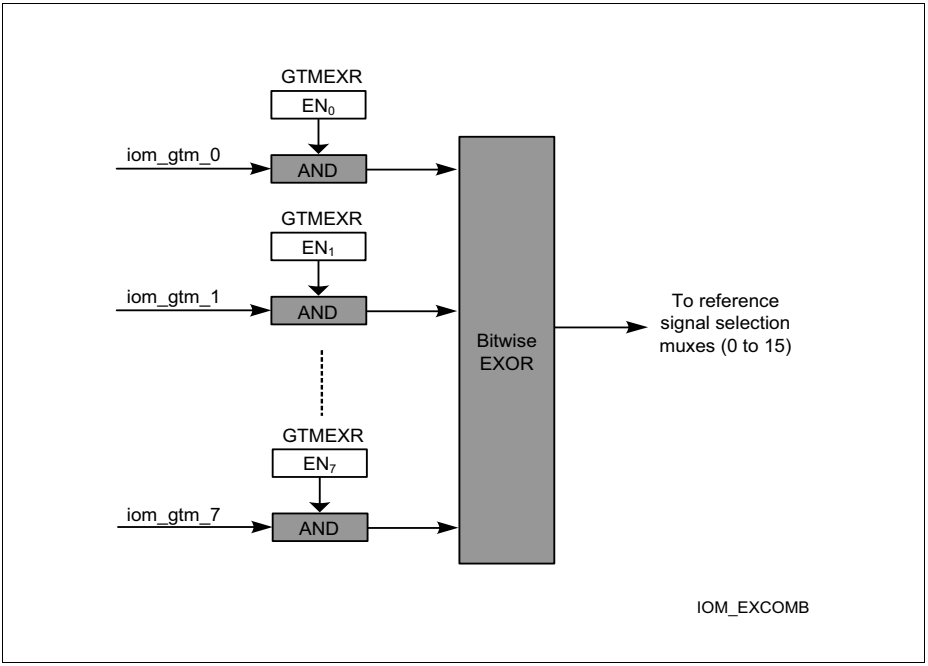


Figure 28-9 EXOR Combiner block diagram

EXOR Combiner Registers

The following registers are assigned to the block:

- GTMEXR = GTM Input EXOR Combiner Selection Configuration Register.

EXOR Combiner Input Signals

- 8 inputs from the GTM module(s).

EXOR Combiner Output Signals

- 1 output signal reflecting the combined logical EXOR function on the associated (and enabled input(s)).

28.7 Logic Analyzer Module (LAM) Description

The IOM instances 16 Logic analyzer Modules (LAM's) for the purposes of monitor and reference signal comparison. Each block accepts one monitor and one reference level signal from the 16 monitor & 16 reference signals available, from the FPC/Input selection block. The behaviour of each LAM is set by configuring the associated register settings.

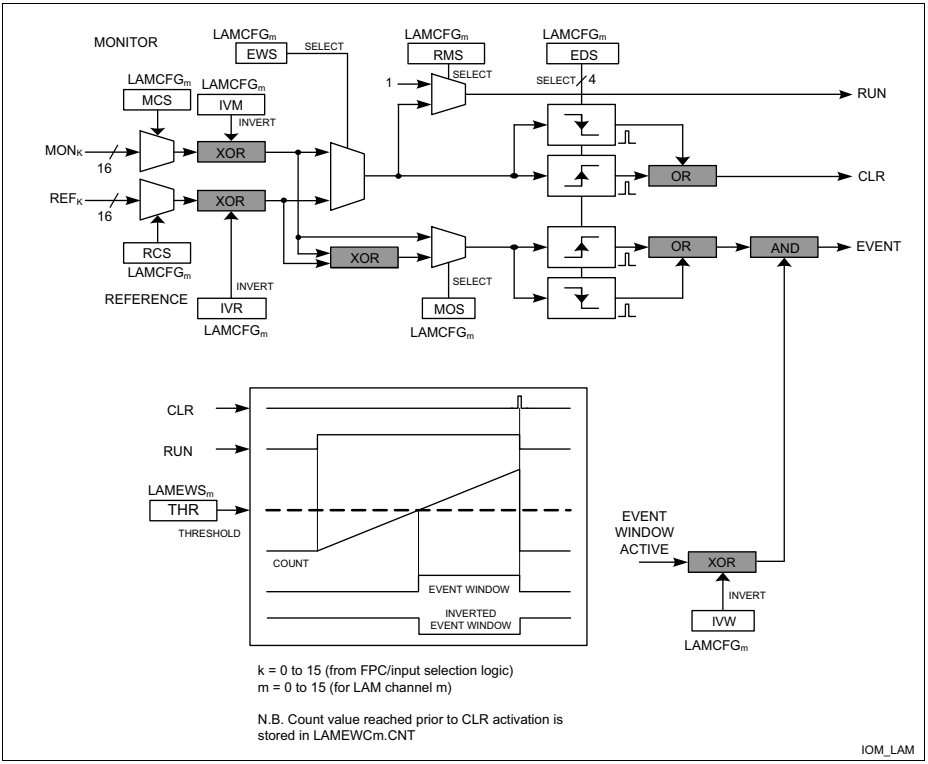


Figure 28-10 Logic Analyzer Module (LAM) block diagram

As shown in **Figure 28-9**, each Logic Analyzer Module features inputs for the Reference & Monitor points, fed from the requisite Filter & Prescaler block. Depending upon the Logic Analyzer Module configuration register (LAMCFG), and the event window settings register (LAMEWS), the block may be configured to monitor a particular type of signal behaviour, input from sensors external to the device in order to monitor, govern and/or control external application behaviour with particular regard to any given safety requirements.

LAM Registers

The following registers are assigned to the Logic Analyzer Module LAM_m (m = 0-15):

- $LAMCFG_m$ = Logic Analyzer Module Configuration Register m
- $LAMEWS_m$ = Logic Analyzer Module Event Window Settings Register m
- $LAMEWC_m$ = Logic Analyzer Module Event Window Count Status Register m

LAM Input Signals

Two inputs exist for each LAM_m:

- Reference signal input, fed from the requisite Filter & Prescaler block.
- Monitor signal input, fed from the requisite Filter & Prescaler block.

Depending upon the required function, both inputs or just the Monitor input may be utilised.

LAM Output Signals

One output is provided by each LAM:

- Event trigger output, configured to be active when a monitored signal falls outside the intended timed and/or periodic behaviour.

28.8 Event Combiner Module (ECM) Description

The IOM instances one Event Combiner Module, which takes the event outputs from each of the 16 Logic Analyzer Modules (LAM's). Some or all of these events can then be combined in a variety of configurable ways in order to generate a single system event.

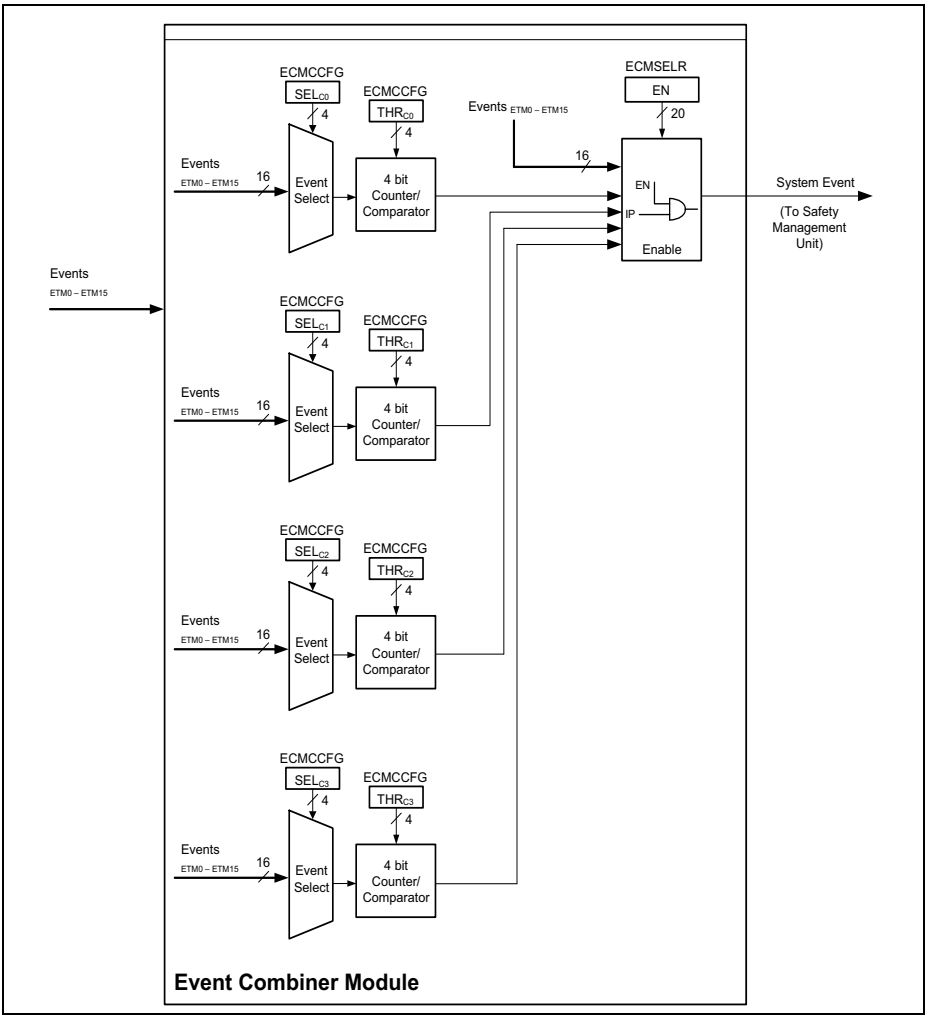


Figure 28-11 Event Combiner Module (ECM) block diagram

Four separate 4bit counters with configurable thresholds are also available to be assigned to count multiple occurring events from any of the 16 available LAM's. The output of these counter units (active once a count threshold has been met/surpassed) can also be included (via configuration) within the generation of the system event. (Note: the alarm output will be connected to the Safety Management Unit, SMU.EMM).

The ECM also includes two Event Trigger History (ETMETH0/1) registers, updated with each system event, used to record the status of the 16 LAM event outputs, thereby identifying which triggers were responsible.

In addition, the ECM incorporates a System Peripheral Bus (SPB) interface for writing configuration and reading status registers appertaining to all subblock within the IOM.

ECM Registers

The following registers are assigned to the ECM:

- ECMCCFG = ECM counter configuration register (for the four 4bit local event counters).
- ECMSELR = ECM global event selection register (for the selection of local events, or multiples thereof, required to generate the global event signal).

ECM Input Signals

- From Logic Analyzer Modules (LAM's)
  - Local event triggers from LAM blocks (0 to 15).
- System Peripheral Bus.

ECM Output Signals

One output is generated by each LAM:

- Global event trigger output, intended to be connected to the Safety Management Unit.
- System Peripheral Bus.

28.9 IOM Registers

This section describes the kernel registers of the IOM unit.

All read and write accesses to an undefined address location within the IOM address space will result in a bus error on the System Peripheral Bus (SPB).

IOM Kernel Register Overview

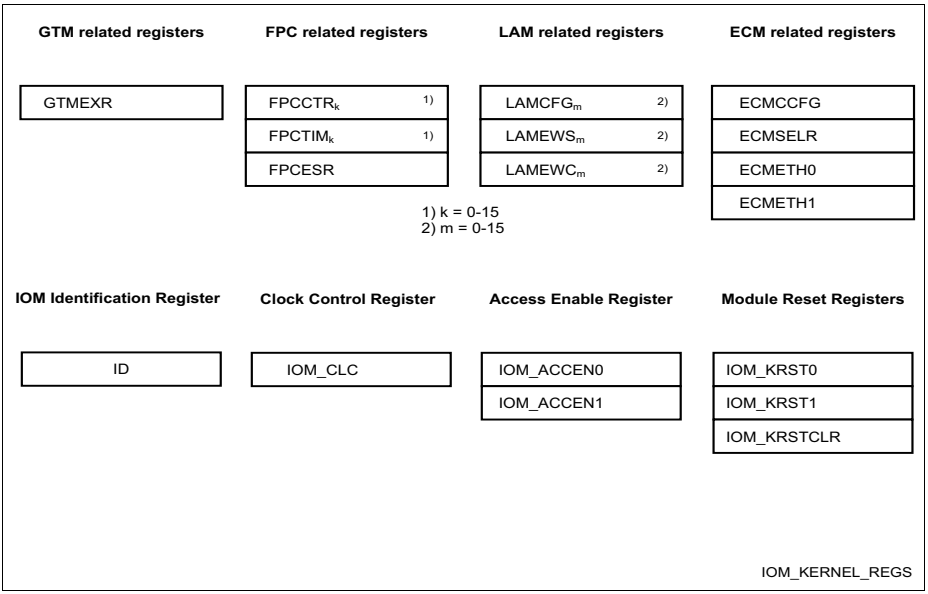


Figure 28-12 IOM Kernel Registers

In the TC21x/TC22x/TC23x, the registers of the IOM units are located in the following address ranges.

Table 28-1 Registers Address Space

Module	Base Address	End Address	Note
IOM	$F003\ 5000_H$	$F003\ 51FF_H$	-

Table 28-2 Registers Overview - IOM Kernel Registers

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Reset Class	Description see
			Read	Write		
IOM_CLC	Clock Control Register	000_H	U, SV	SV, E, P	3	Page 37
ID	IOM Identification Register	008_H	U, SV	BE	3	Page 20
IOM_KRSTCLR	Reset Status Clear Register	$01C_H$	U, SV	SV, E, P	3	Page 43
IOM_KRST1	Reset Control Register 1	020_H	U, SV	SV, E, P	3	Page 42
IOM_KRST0	Reset Control Register 0	024_H	U, SV	SV, E, P	3	Page 40
IOM_ACCEN1	Access Enable Register 1	028_H	U, SV	SV, SE	3	Page 39
IOM_ACCEN0	Access Enable Register 0	$02C_H$	U, SV	SV, SE	3	Page 38
ECMCCFG	Event Combiner Module Counter Configuration Register	030_H	U, SV	U, SV	3	Page 32
ECMSELR	Event Combiner Module Global Event Selection Register	034_H	U, SV	U, SV	3	Page 34
ECMETH0	Event Trigger History Register 0	038_H	U, SV	U, SV	3	Page 35
ECMETH1	Event Trigger History Register 1	$03C_H$	U, SV	U, SV	3	Page 36
GTMEXR	GTM Input EXOR Combiner Selection Register	040_H	U, SV	U, SV	3	Page 25
FPCESR	Filter and Prescaler Cells Rising & Falling Edge Status Register	078_H	U, SV	U, SV	3	Page 21
$FPCCTR_k$	Filter and Prescaler Cell Control Register k (k = 0-15)	$080_H + k * 4$)	U, SV	U, SV	3	Page 22

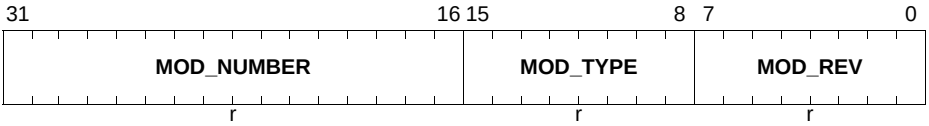
Table 28-2 Registers Overview - IOM Kernel Registers

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Reset Class	Description see
			Read	Write		
$FPCTIM_k$	Filter and Prescaler Cell Timer Register k (k = 0-15)	$0C0_H$ + k * 4)	U, SV	U, SV	3	Page 24
$LAMEWC_m$	Logic Analyzer Module Event Window Count Status Register m (m = 0-15)	100_H + m * 4	U, SV	BE	3	Page 27
$LAMCFG_m$	Logic Analyzer Module Configuration Register m (m = 0-15)	180_H + m * 4	U, SV	U, SV	3	Page 28
$LAMEWS_m$	Logic Analyzer Module Event Window Setting Register m (m = 0-15)	$1C0_H$ + m * 4	U, SV	U, SV	3	Page 31

28.9.1 IOM Identification Register (IOM_ID)

The IOM Identification Register ID contains read-only information about the module version.

IOM_ID
IOM Identification Register **(008_H)** **Reset Value: $00CC\ C001_H$**



Field	Bits	Type	Description
MOD_REV	[7:0]	r	**Module Revision Number** MOD_REV defines the Module revision number. The value of a module revision starts with 01_H (first revision).
MOD_TYPE	[15:8]	r	**Module Number Value** This bit field defines the module as a 32 bit module: $C0_H$
MOD_NUM	[31:16]	r	**Module Number Value** This bit field defines the identification number for the IOM.

28.9.2 Filter & Prescaler Cell (FPC) Registers

IOM Filter and Prescaler Cells Rising & Falling Edge Status Register (IOM_FPCESR)

The Filter and Prescaler Edge Status Register stores the state of detected rising and falling edges from each of the FPC cells.

The register can be written to selectively clear individual bits.

Individual bits are also cleared with a write to the control register (FPCCTRk) or timer register (FPCTIMk).

IOM_FPCESR
IOM Filter and Prescaler Cells Rising & Falling Edge Status Register
(078_H) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG 31	**REG 30**	**REG 29**	**REG 28**	**REG 27**	**REG 26**	**REG 25**	**REG 24**	**REG 23**	**REG 22**	**REG 21**	**REG 20**	**REG 19**	**REG 18**	**REG 17**	**REG 16**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

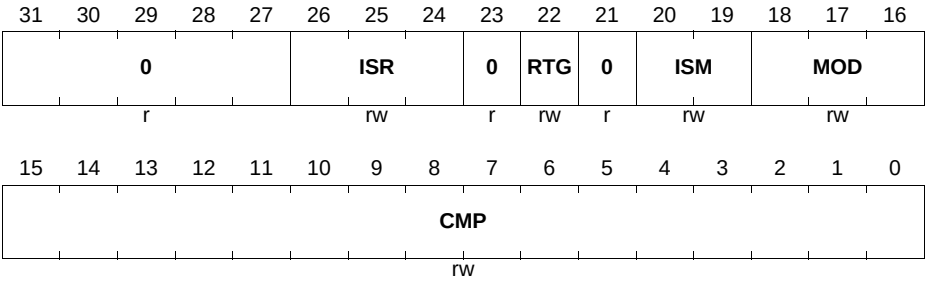
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEG 15	**FEG 14**	**FEG 13**	**FEG 12**	**FEG 11**	**FEG 10**	**FEG 9**	**FEG 8**	**FEG 7**	**FEG 6**	**FEG 5**	**FEG 4**	**FEG 3**	**FEG 2**	**FEG 1**	**FEG 0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
FEGk (k = 15-0)	k	rwh	**Falling Edge Glitch Flag for FPCk** 0_B No falling edge of glitch detected during filtering 1_B Falling edge of glitch detected during filtering
REGk (k = 15-0)	16+k	rwh	**Rising Edge Glitch Flag for FPCk** 0_B No rising edge of glitch detected during filtering 1_B Rising edge of glitch detected during filtering

IOM filter & Prescaler Cell Control register (IOM_FPCCTRk)

The IOM Filter & Prescaler Control Register is used to configure the FPC and mux selection logic.

IOM_FPCCTRk (k = 0-15)
IOM Filter and Prescaler Cell Control Register$_k$
(080_H+k*4_H) **Reset Value: 0000 0000_H**



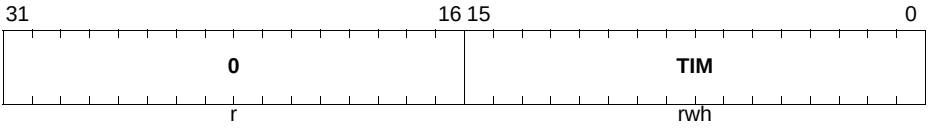
Field	Bits	Type	Description
CMP	[15:0]	rw	**Threshold Value of Filter & Prescaler Cell k** CMP is the 16-bit threshold value that is compared with the 16-bit timer value FPCTIMk.TIM.
MOD	[18:16]	rw	**Operation Mode Selection for Filter & Prescaler Cell k** 000_B Delayed Debounce Filter Mode on both edges 001_B Immediate Debounce Filter Mode on both edges 010_B Rising edge: Immediate Debounce Filter Mode, falling edge: no filtering 011_B Rising edge: no filtering, falling edge: Immediate Debounce Filter Mode 100_B Rising edge: Delayed Debounce Filter Mode, falling edge: Immediate Debounce Filter Mode 101_B Rising edge: Immediate Debounce Filter Mode, falling edge: Delayed Debounce Filter Mode 110_B Prescaler Mode (triggered on rising edge) 111_B Prescaler Mode (triggered on falling edge)

Field	Bits	Type	Description
ISM	[20:19]	rw	**Monitor Input Signal Selection for Filter & Prescaler Cell k** IPS determines the signal input used for edge detection. 00_B Signal input Pn_IN (from portlogic) selected 01_B Monitor Signal Input 0 selected 10_B Monitor Signal Input 1 selected 11_B Monitor Signal Input 2 selected
0	21	r	**Reserved** Read as 0; should be written with 0.
RTG	22	rw	**Reset Timer behaviour for Filter & Prescaler Cell k on Glitch** 0_B Timer for FPCk is decremented on glitch 1_B Timer for FPCk is cleared on glitch This bit is effective in Delayed Debounce Filter Mode only.
0	23	r	**Reserved** Read as 0; should be written with 0.
ISR	[26:24]	rw	**Reference Input Signal Selection for Filter & Prescaler Cell k** ISM determines the input used for the channel k reference signal. 000_B Signal input Pn_IN (from portlogic) selected 001_B Reference Signal Input 0selected 010_B Reference Signal Input 1selected 011_B Reference Signal Input 2selected $1xx_B$ GTM XOR Combiner selected
0	[31:27]	r	**Reserved** Read as 0; should be written with 0.

IOM Filter & Prescaler Cell k Timer Register (IOM_FPCTIMk)

The IOM Filter & Prescaler Cell Timer Register is used to set the value of the timer for the required application.

IOM_FPCTIMk (k = 0-15)
IOM Filter and Prescaler Cell kTimer Register$_k$
($0C0_H$+k*4_H) **Reset Value: 0000 0000_H**



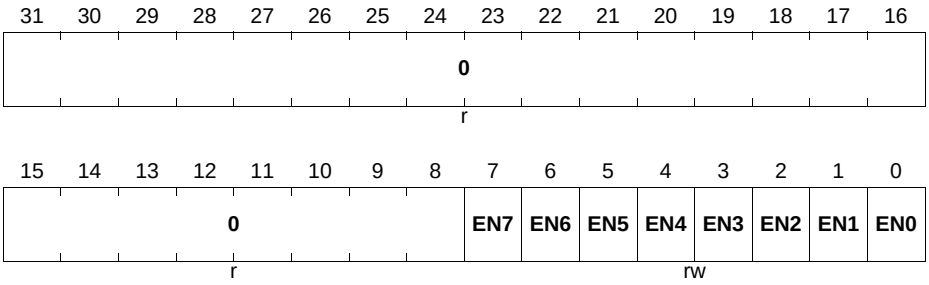
Field	Bits	Type	Description
TIM	[15:0]	rwh	**Timer Value of Filter and Prescaler Cell k** N.B. a write of any value to the TIM bitfield will result in it's contents being set to the reset value. This is also undertaken whenever the IOM_FPCCTRk register is written with a new value.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

28.9.3 GTM Input Related Registers

IOM EXOR Combiner GTM signal input Selection Register (IOM_GTMEXR)

Enables the incoming GTM-connected input signals for inclusion within a combined EXOR function.

IOM_GTMEXR
IOM GTM Input EXOR Combiner Selection Register
(040_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
EN0	0	rw	**GTM input 0 selection for EXOR combiner** 0_B Input not selected for EXOR combiner. 1_B Input selected for EXOR combiner.
EN1	1	rw	**GTM input 1 selection for EXOR combiner** 0_B Input not selected for EXOR combiner. 1_B Input selected for EXOR combiner.
EN2	2	rw	**GTM input 2 selection for EXOR combiner** 0_B Input not selected for EXOR combiner. 1_B Input selected for EXOR combiner.
EN3	3	rw	**GTM input 3 selection for EXOR combiner** 0_B Input not selected for EXOR combiner. 1_B Input selected for EXOR combiner.
EN4	4	rw	**GTM input 4 selection for EXOR combiner** 0_B Input not selected for EXOR combiner. 1_B Input selected for EXOR combiner.
EN5	5	rw	**GTM input 5 selection for EXOR combiner** 0_B Input not selected for EXOR combiner. 1_B Input selected for EXOR combiner.

Field	Bits	Type	Description
EN6	6	rw	**GTM input 6 selection for EXOR combiner** 0_B Input not selected for EXOR combiner. 1_B Input selected for EXOR combiner.
EN7	7	rw	**GTM input 7 selection for EXOR combiner** 0_B Input not selected for EXOR combiner. 1_B Input selected for EXOR combiner.
0	[31:8]	r	**Reserved** Read as 0; should be written with 0.

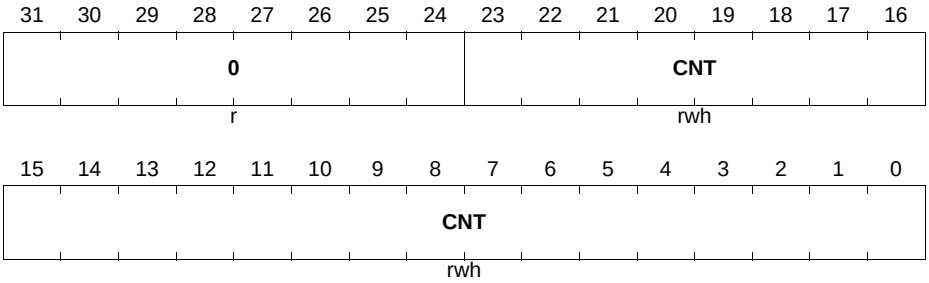
28.9.4 Logic Analyzer Module (LAM) Registers

IOM Logic Analyzer Module Event Window Count Status register (IOM_LAMEWCm)

Used to store the window count value reached prior to being cleared in the LAM block once an event has been generated.

Note: A write to this register will result in a bus error, i.e.existing contents unaffected.

IOM_LAMEWCm (m = 0-15)
IOM Logic Analyzer Module Event Window Count Status Register$_m$
(100_H+m*4_H) **Reset Value: 0000 0000_H**

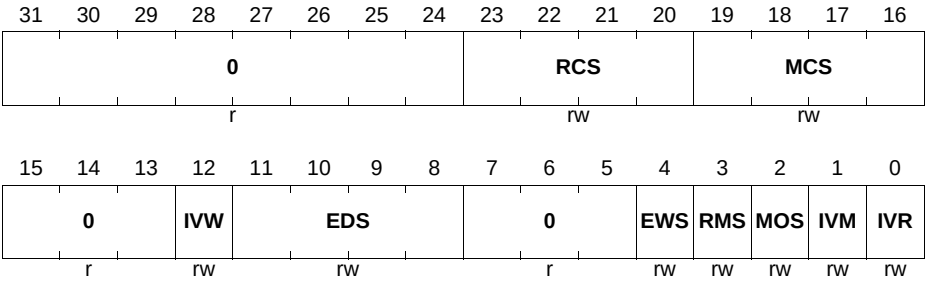


Field	Bits	Type	Description
CNT	[23:0]	r	**Event Window Count Value LAM block m** The count value of the event window attained coincident with an event occuring.
0	[31:24]	r	**Reserved** Read as 0.

IOM Logic Analyzer Module Configuration Register (IOM_LAMCFGm)

Use to configure the application-specific settings for each LAM block, including signal selection muxes at the block inputs.

IOM_LAMCFGm (m = 0-15)
IOM Logic Analyzer Module Configuration Register$_m$
(180_H+m*4_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
IVR	0	rw	**Invert Reference LAM block m** This bit field determines whether the reference signal from the FPCrch is inverted or not. 0_B Don't invert reference signal from FPC. 1_B Invert reference signal from FPC.
IVM	1	rw	**Invert Monitor LAM block m** This bit field determines whether the monitor signal from the FPCmch is inverted or not. 0_B Don't invert monitor signal from FPC. 1_B Invert monitor signal from FPC.
MOS	2	rw	**Monitor Source Select LAM block m** This bit field determines whether the monitor signal from the FPCmch is sourced directly or compared (EXOR'd) with the reference signal from the FPCrch for the event compare. 0_B Monitor signal is sourced directly from FPCmch. 1_B Monitor signal is EXOR'd with FPCrch.

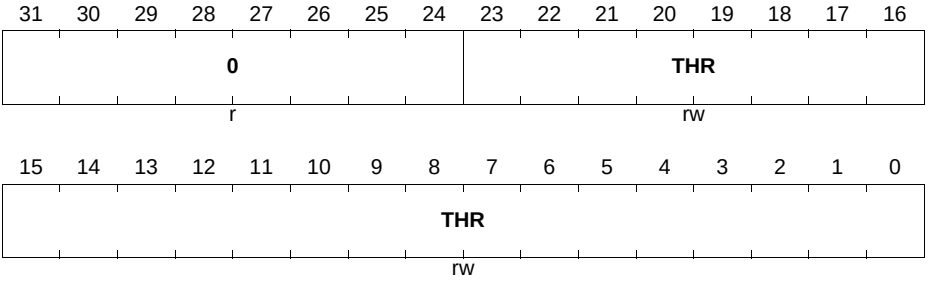
Field	Bits	Type	Description
RMS	3	rw	**Runmode Select LAM block m** This bit field determines whether the event window generation is free-running or gated with the monitor or reference. 0_B Event window generation is free-running. 1_B Event window generation is gated with the monitor or reference signal.
EWS	4	rw	**Event Window Select LAM block m** This bit field determines whether the event window generation is from the monitor or reference signal. 0_B Event window generation is determined from the reference signal. 1_B Event window generation is determined from the monitor signal.
0	[7:5]	r	**Reserved** Read as 0; should be written with 0.
EDS	[11:8]	rw	**Event Window Active Edge Selection LAM block m** This bit field determines which active edges of the monitor and reference signals are used for the event window generation. $xx00_B$ Neither edge used to clear event windowcounter. $xx01_B$ Positive edge used to clear event windowcounter. $xx10_B$ Negative edge used to clear event windowcounter. $xx11_B$ Either edge used to clear event windowcounter. $00xx_B$ Neither edge used to gate event generation. $01xx_B$ Positive edge used to gate event generation. $10xx_B$ Negative edge used to gate event generation. $11xx_B$ Either edge used to gate event generation.
IVW	12	rw	**Invert Event Window LAM block m** This bit field determines whether the event window polarity is inverted or not. 0_B Event window non-inverted. 1_B Event window inverted.
0	[15:13]	r	**Reserved** Read as 0; should be written with 0.

Field	Bits	Type	Description
MCS	[19:16]	rw	**Monitor Input Signal Selection LAM block m** This bit field determines which FPC/mux block k monitor output signal is to be used for LAM block m. 0000_B Monitor signal provided by FPC/mux block 0. 0001_B Monitor signal provided by FPC/mux block 1. 0010_B Monitor signal provided by FPC/mux block 2. 0011_B Monitor signal provided by FPC/mux block 3. 0100_B Monitor signal provided by FPC/mux block 4. 0101_B Monitor signal provided by FPC/mux block 5. 0110_B Monitor signal provided by FPC/mux block 6. 0111_B Monitor signal provided by FPC/mux block 7. 1000_B Monitor signal provided by FPC/mux block 8. 1001_B Monitor signal provided by FPC/mux block 9. 1010_B Monitor signal provided by FPC/mux block 10. 1011_B Monitor signal provided by FPC/mux block 11. 1100_B Monitor signal provided by FPC/mux block 12. 1101_B Monitor signal provided by FPC/mux block 13. 1110_B Monitor signal provided by FPC/mux block 14. 1111_B Monitor signal provided by FPC/mux block 15.
RCS	[23:20]	rw	**Reference Input Signal Selection LAM block m** This bit field determines which FPC/mux block k reference output signal is to be used for LAM block m. 0000_B Reference signal provided by FPC/mux block 0. 0001_B Reference signal provided by FPC/mux block 1. 0010_B Reference signal provided by FPC/mux block 2. 0011_B Reference signal provided by FPC/mux block 3. 0100_B Reference signal provided by FPC/mux block 4. 0101_B Reference signal provided by FPC/mux block 5. 0110_B Reference signal provided by FPC/mux block 6. 0111_B Reference signal provided by FPC/mux block 7. 1000_B Reference signal provided by FPC/mux block 8. 1001_B Reference signal provided by FPC/mux block 9. 1010_B Reference signal provided by FPC/mux block 10. 1011_B Reference signal provided by FPC/mux block 11. 1100_B Reference signal provided by FPC/mux block 12. 1101_B Reference signal provided by FPC/mux block 13. 1110_B Reference signal provided by FPC/mux block 14. 1111_B Reference signal provided by FPC/mux block 15.
0	[31:24]	r	**Reserved** Read as 0; should be written with 0.

IOM Logic Analyzer Module Event Window Configuration Register (IOM_LAMEWSm)

Used to set the threshold value for the event window counter.

IOM_LAMEWSm (m = 0-15)
IOM Logic Analyzer Module Event Window Configuration Register$_m$
($1C0_H$+m*4_H) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
THR	[23:0]	rw	**Event Window Count Threshold** This bit field determines the threshold value for the event window counter from which an event is generated. Note: if THR has a value of 0 then no event will be generated.
0	[31:24]	r	**Reserved** Read as 0; should be written with 0.

28.9.5 Event Combiner Module (ECM) Registers

Event Combiner Module Counter Configuration register (IOM_ECMCCFG)

The Event Combiner Configuration register is used to configure each of the four event counter submodules, enabling multiple events from one selected LAM block (for each event counter submodule) to be counted in the generation of a system event.

IOM_ECMCCFG
IOM Event Combiner Module Counter Configuration Register
(030_H) Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
THRC3				SELC3				THRC2				SELC2			
rw				rw				rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
THRC1				SELC1				THRC0				SELC0			
rw				rw				rw				rw			

Field	Bits	Type	Description
SELC0, SELC1, SELC2, SELC3	[3:0], [11:8], [19:16], [27:24]	rw	**Event Channel Select** This bit field determines which channel event output is to be routed to counter$_{Cn}$. 0000_B Select channel 0 event output to be counted. 0001_B Select channel 1 event output to be counted. 0010_B Select channel 2 event output to be counted. 0011_B Select channel 3 event output to be counted. 0100_B Select channel 4 event output to be counted. 0101_B Select channel 5 event output to be counted. 0110_B Select channel 6 event output to be counted. 0111_B Select channel 7 event output to be counted. 1000_B Select channel 8 event output to be counted. 1001_B Select channel 9 event output to be counted. 1010_B Select channel 10 event output to be counted. 1011_B Select channel 11 event output to be counted. 1100_B Select channel 12 event output to be counted. 1101_B Select channel 13 event output to be counted. 1110_B Select channel 14 event output to be counted. 1111_B Select channel 15 event output to be counted.
THRC0, THCR1, THCR2, THCR3	[7:4], [15:12], [23:20], [31:28]	rw	**Channel Event Counter Threshold** This bit field determines the threshold count value. Upon the counter meeting the threshold, the counter event output becomes active high for one clock cycle and the count is reset to zero. 0000_B Counter disabled (counter event output set to inactive, despite any incoming channel events). 0001_B Minimum threshold count value. ... 1111_B Maximum threshold count value.

Event Combiner Module Global Event Selection Register (IOM_ECMSELR)

Configured to combine individual and multiple (counted) events from the LAM modules in order to generate a global (system) event.

IOM_ECMSELR
IOM Event Combiner Module Global Event Selection Register
(034_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0												CTS 3	CTS 2	CTS 1	CTS 0
r												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CES 15	CES 14	CES 13	CES 12	CES 11	CES 10	CES 9	CES 8	CES 7	CES 6	CES 5	CES 4	CES 3	CES 2	CES 1	CES 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CESn (n=0-15)	n	rw	**Event Combiner Selection** The setting of individual bitfields determines the inclusion of the respective channel event in the generation of the global event (OR function). 0_B Don't include channel event/event counter output within the global event generation. 1_B Include channel event/event counter output within the global event generation.
CTSn (n=0-3)	n+16	rw	**Accumulated (Counted) Event Combiner Selection** The setting of individual bitfields determines the inclusion of the respective channel event counter output (1 of 4) in the generation of the global event (AND function). 0_B Don't include channel event/event counter output within the global event generation. 1_B Include channel event/event counter output within the global event generation.
0	[31:20]	r	**Reserved** Read as 0; should be written with 0.

Event Combiner Module Event Trigger History Register (IOM_ECMETH0)

One of two registers (IOM_ECMETH0 and IOM_ECMETH1) that record the status of LAM's 0 to 15 event outputs for the last 4 generated system events.

The generation of a system event will cause the status of the event outputs from LAM blocks 0 to 15 to be recorded in ETA0-15, respectively. Just prior to this happening, the previous status held in ETA0-15 is moved to ETB0-15. Similarly, the previous contents of ETB0-15 and ETC0-15 are moved to ETC0-15 and ETD15-0, respectively. The old contents of ETD15-0, being lost.

At reset (or a register write to either IOM_ECMETH0 or IOM_ECMETH1), all Event Trigger Active flags (ETA, ETB, ETC, ETD) will be cleared.

IOM_ECMETH0
IOM Event Combiner Module Event Trigger History Register 0
(038_H) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETB 15	**ETB 14**	**ETB 13**	**ETB 12**	**ETB 11**	**ETB 10**	**ETB 9**	**ETB 8**	**ETB 7**	**ETB 6**	**ETB 5**	**ETB 4**	**ETB 3**	**ETB 2**	**ETB 1**	**ETB 0**
rwh															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETA 15	**ETA 14**	**ETA 13**	**ETA 12**	**ETA 11**	**ETA 10**	**ETA 9**	**ETA 8**	**ETA 7**	**ETA 6**	**ETA 5**	**ETA 4**	**ETA 3**	**ETA 2**	**ETA 1**	**ETA 0**
rwh															

Field	Bits	Type	Description
ETAn (n=0-15)	n	rwh	**LAM 0-15 Event Trigger Activity (last)** Contains the status of the event trigger outputs for each of the LAM blocks for the last generated event(s).
ETBn (n=0-15)	n+16	rwh	**LAM 0-15 Event Trigger Activity (previous ETA0-15)** Contains the previous contents of ETA0-15 prior to being updated.

Event Combiner Module Event Trigger History Register (IOM_ECMETH1)

One of two registers (IOM_ECMETH0 and IOM_ECMETH1) that record the status of LAM's 0 to 15 event outputs for the last 4 generated system events.

The generation of a system event will cause the status of the event outputs from LAM blocks 0 to 15 to be recorded in ETA0-15, respectively. Just prior to this happening, the previous status held in ETA0-15 is moved to ETB0-15. Similarly, the previous contents of ETB0-15 and ETC0-15 are moved to ETC0-15 and ETD15-0, respectively. The old contents of ETD15-0, being lost.

At reset (or a register write to either IOM_ECMETH0 or IOM_ECMETH1), all Event Trigger Active flags (ETA, ETB, ETC, ETD) will be cleared.

IOM_ECMETH1
IOM Event Combiner Module Event Trigger History Register 1
($03C_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETD 15	**ETD 14**	**ETD 13**	**ETD 12**	**ETD 11**	**ETD 10**	**ETD 9**	**ETD 8**	**ETD 7**	**ETD 6**	**ETD 5**	**ETD 4**	**ETD 3**	**ETD 2**	**ETD 1**	**ETD 0**
rwh															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETC 15	**ETC 14**	**ETC 13**	**ETC 12**	**ETC 11**	**ETC 10**	**ETC 9**	**ETC 8**	**ETC 7**	**ETC 6**	**ETC 5**	**ETC 4**	**ETC 3**	**ETC 2**	**ETC 1**	**ETC 0**
rwh															

Field	Bits	Type	Description
ETCn (n=0-15)	n	rwh	**LAM 0-15 Event Trigger Activity (previous ETB0-15)** Contains the previous contents of ETB0-15 prior to being updated.
ETDn (n=0-15)	n+16	rwh	**LAM 0-15 Event Trigger Activity (previous ETC0-15)** Contains the previous contents of ETC0-15 prior to being updated.

28.9.6 System Registers

IOM Clock Control Register (IOM_CLC)

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI_FPI for the module. Where a module kernel is connected to the CLC clock control interface, CLC controls the $f_{<>}$ module clock signal, sleep mode and disable mode for the module.

IOM_CLC
IOM Clock Control Register (000_H) Reset Value: 0000 0003$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RMC								0				EDIS	0	DISS	DISR
rw								r				rw	rw	rh	rw

Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module.
0	2	rw	**Reserved** Read as 0; should be written with 0.
EDIS	3	rw	**Sleep Mode Enable Control** Used to control module's sleep mode.
RMC	[15:8]	rw	**8-bit Clock Divider Value in RUN Mode**
0	[31:16], [7:4]	r	**Reserved** Read as 0; should be written with 0.

Note: The number of module clock cycles (wait states) which are required by the kernel to execute a read or write access depends on the selected CLC clock frequency, which is selected via bit field RMC in the CLC register. Therefore, increasing

CLC.RMC may result in a longer FPI Bus read cycle access time for kernel registers and can also slow down the write throughput to the kernel registers.

IOM Access Enable Register (IOM_ACCEN0)

The Access Enable Register 0 controls write access[1)] for transactions with the on chip bus master TAG ID 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000_B, EN1 -> TAG ID 000001_B, ... , EN31 -> TAG ID 011111_B.

IOM_ACCEN0
IOM Access Enable Register 0 ($02C_H$) Reset Value: FFFF FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN 31	**EN 30**	**EN 29**	**EN 28**	**EN 27**	**EN 26**	**EN 25**	**EN 24**	**EN 23**	**EN 22**	**EN 21**	**EN 20**	**EN 19**	**EN 18**	**EN 17**	**EN 16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN 15	**EN 14**	**EN 13**	**EN 12**	**EN 11**	**EN 10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n = 0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

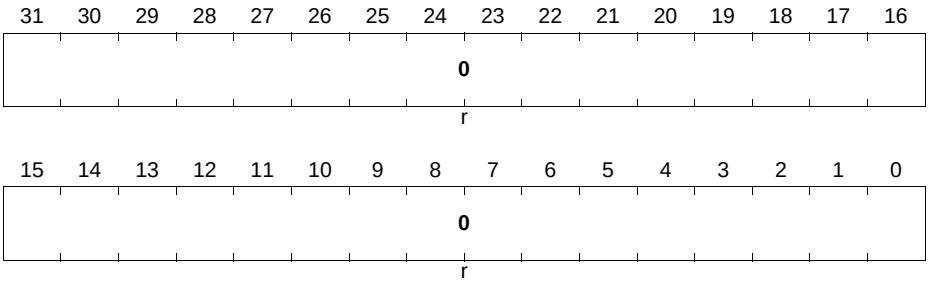
1) The BPI_FPI Access Enable functionality controls only write transactions to the CLC, OCS, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers

IOM Access Enable Register (IOM_ACCEN1)

The Access Enable Register 1 controls write access[1)] for transactions with the on chip bus master TAG ID 100000_B to 111111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the AURIX devices.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001_B, ... , EN31 -> TAG ID 111111_B.

IOM_ACCEN1
IOM Access Enable Register 1 (028_H) Reset Value: 0000 0000_H



Field	Bits	Type	Description
0	[31:0]	r	**Reserved** Read as 0; should be written with 0.

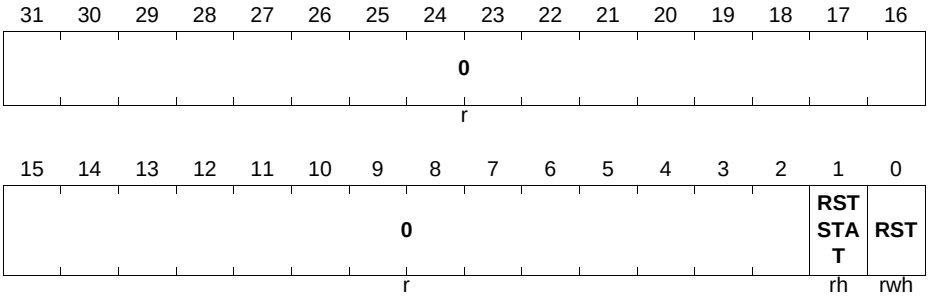
IOM Kernel Reset Register 0 (IOM_KRST0)

The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers.The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the KRSTCLE.CLR register bit.

Note: During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

IOM_KRST0
IOM Kernel Reset Register 0 (024_H) Reset Value: 0000 0000$_H$



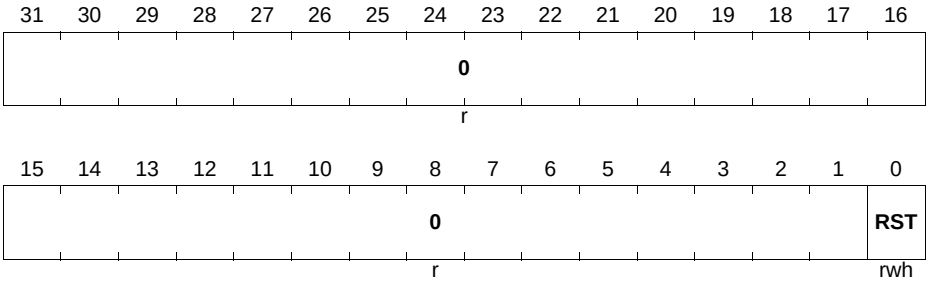
Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed.

Field	Bits	Type	Description
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. 0_B No kernel reset was executed 1_B Kernel reset was executed This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register.
0	[31:2]	r	**Reserved** Read as 0; should be written with 0.

IOM Kernel Reset Register 1 (IOM_KRST1)

The Kernel Reset Register 1 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset registers. The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

IOM_KRST1
IOM Kernel Reset Register 1 (020_H) Reset Value: 0000 0000_H

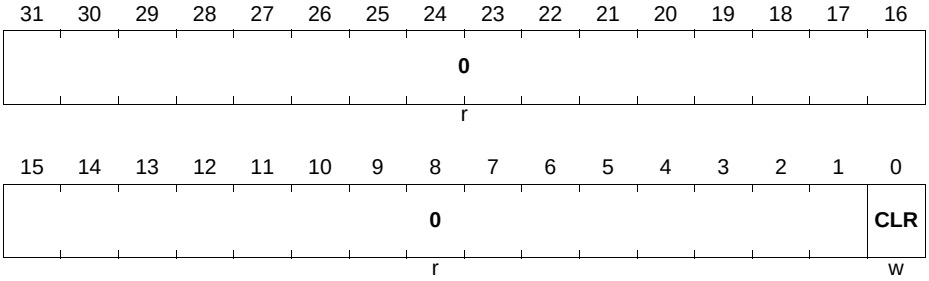


Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

IOM Kernel Reset Status Clear Register (IOM_KRSTCLR)

The Kernel Reset Status Clear register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

IOM_KRSTCLR
IOM Kernel Reset Status Clear Register($01C_H$) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT Read always as 0.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

28.10 SoC Integration

The following tables describes how the IOM Monitor & Reference signals are connected to the portlogic.

IOM Signal Name	Pinning Table Signal	Description	Selection setting
iom_pin_i(0)	IOM_PIN0	GPIO pad input	IOM_FPCCTR$_0$.ISM=00
iom_pin_i(1)	IOM_PIN1	GPIO pad input	IOM_FPCCTR$_1$.ISM=00
iom_pin_i(2)	IOM_PIN2	GPIO pad input	IOM_FPCCTR$_2$.ISM=00
iom_pin_i(3)	IOM_PIN3	GPIO pad input	IOM_FPCCTR$_3$.ISM=00
iom_pin_i(4)	IOM_PIN4	GPIO pad input	IOM_FPCCTR$_4$.ISM=00
iom_pin_i(5)	IOM_PIN5	GPIO pad input	IOM_FPCCTR$_5$.ISM=00
iom_pin_i(6)	IOM_PIN6	GPIO pad input	IOM_FPCCTR$_6$.ISM=00
iom_pin_i(7)	IOM_PIN7	GPIO pad input	IOM_FPCCTR$_7$.ISM=00
iom_pin_i(8)	IOM_PIN8	GPIO pad input	IOM_FPCCTR$_8$.ISM=00
iom_pin_i(9)	IOM_PIN9	GPIO pad input	IOM_FPCCTR$_9$.ISM=00
iom_pin_i(10)	IOM_PIN10	GPIO pad input	IOM_FPCCTR$_{10}$.ISM=00
iom_pin_i(11)	IOM_PIN11	GPIO pad input	IOM_FPCCTR$_{11}$.ISM=00
iom_pin_i(12)	IOM_PIN12	GPIO pad input	IOM_FPCCTR$_{12}$.ISM=00
iom_pin_i(13)	IOM_PIN13	GPIO pad input	IOM_FPCCTR$_{13}$.ISM=00
iom_pin_i(14)	IOM_PIN14	GPIO pad input	IOM_FPCCTR$_{14}$.ISM=00
iom_pin_i(15)	IOM_PIN15	GPIO pad input	IOM_FPCCTR$_{15}$.ISM=00
iom_mon_0_i(0)	IOM_MON0_0	GTM output: TOUT22	IOM_FPCCTR$_0$.ISM=01
iom_mon_0_i(1)	IOM_MON0_1	GTM output: TOUT23	IOM_FPCCTR$_1$.ISM=01
iom_mon_0_i(2)	IOM_MON0_2	GTM output: TOUT24	IOM_FPCCTR$_2$.ISM=01
iom_mon_0_i(3)	IOM_MON0_3	GTM output: TOUT25	IOM_FPCCTR$_3$.ISM=01
iom_mon_0_i(4)	IOM_MON0_4	GTM output: TOUT26	IOM_FPCCTR$_4$.ISM=01
iom_mon_0_i(5)	IOM_MON0_5	GTM output: TOUT27	IOM_FPCCTR$_5$.ISM=01
iom_mon_0_i(6)	IOM_MON0_6	GTM output: TOUT28	IOM_FPCCTR$_6$.ISM=01
iom_mon_0_i(7)	IOM_MON0_7	GTM output: TOUT29	IOM_FPCCTR$_7$.ISM=01
iom_mon_0_i(8)	IOM_MON0_8	GTM output: TOUT30	IOM_FPCCTR$_8$.ISM=01
iom_mon_0_i(9)	IOM_MON0_9	GTM output: TOUT31	IOM_FPCCTR$_9$.ISM=01
iom_mon_0_i(10)	IOM_MON0_10	GTM output: TOUT32	IOM_FPCCTR$_{10}$.ISM=01
iom_mon_0_i(11)	IOM_MON0_11	GTM output: TOUT33	IOM_FPCCTR$_{11}$.ISM=01
iom_mon_0_i(12)	IOM_MON0_12	GTM output: TOUT34	IOM_FPCCTR$_{12}$.ISM=01
iom_mon_0_i(13)	IOM_MON0_13	GTM output: TOUT68	IOM_FPCCTR$_{13}$.ISM=01
iom_mon_0_i(14)	IOM_MON0_14	GTM output: TOUT69	IOM_FPCCTR$_{14}$.ISM=01
iom_mon_0_i(15)	IOM_MON0_15	GTM output: TOUT70	IOM_FPCCTR$_{15}$.ISM=01
iom_mon_1_i(0)	IOM_MON1_0	CCU60 output: CC62	IOM_FPCCTR$_0$.ISM=10
iom_mon_1_i(1)	IOM_MON1_1	CCU60 output: CC61	IOM_FPCCTR$_1$.ISM=10
iom_mon_1_i(2)	IOM_MON1_2	CCU60 output: CC60	IOM_FPCCTR$_2$.ISM=10
iom_mon_1_i(3)	IOM_MON1_3	CCU60 output: COUT60	IOM_FPCCTR$_3$.ISM=10
iom_mon_1_i(4)	IOM_MON1_4	CCU60 output: COUT61	IOM_FPCCTR$_4$.ISM=10
iom_mon_1_i(5)	IOM_MON1_5	CCU60 output: COUT62	IOM_FPCCTR$_5$.ISM=10
iom_mon_1_i(6)	IOM_MON1_6	CCU60 output: COUT63	IOM_FPCCTR$_6$.ISM=10
iom_mon_1_i(7)	IOM_MON1_7	CCU61 output: COUT63	IOM_FPCCTR$_7$.ISM=10
iom_mon_1_i(8)	IOM_MON1_8	CCU61 output: CC60	IOM_FPCCTR$_8$.ISM=10
iom_mon_1_i(9)	IOM_MON1_9	CCU61 output: CC61	IOM_FPCCTR$_9$.ISM=10
iom_mon_1_i(10)	IOM_MON1_10	CCU61 output: CC62	IOM_FPCCTR$_{10}$.ISM=10
iom_mon_1_i(11)	IOM_MON1_11	CCU61 output: COUT60	IOM_FPCCTR$_{11}$.ISM=10
iom_mon_1_i(12)	IOM_MON1_12	CCU61 output: COUT61	IOM_FPCCTR$_{12}$.ISM=10
iom_mon_1_i(13)	IOM_MON1_13	CCU61 output: COUT62	IOM_FPCCTR$_{13}$.ISM=10
iom_mon_1_i(14)	IOM_MON1_14	GTM output: TOUT61	IOM_FPCCTR$_{14}$.ISM=10
iom_mon_1_i(15)	IOM_MON1_15	GTM output: TOUT62	IOM_FPCCTR$_{15}$.ISM=10
iom_mon_2_i(0)	IOM_MON2_0	QSPI0 output: MRST0	IOM_FPCCTR$_0$.ISM=11
iom_mon_2_i(1)	IOM_MON2_1	QSPI1 output: MRST1	IOM_FPCCTR$_1$.ISM=11
iom_mon_2_i(2)	IOM_MON2_2	QSPI2 output: MRST2	IOM_FPCCTR$_2$.ISM=11
iom_mon_2_i(3)	IOM_MON2_3	QSPI3 output: MRST3	IOM_FPCCTR$_3$.ISM=11
iom_mon_2_i(4)	IOM_MON2_4	GTM output: TOUT63	IOM_FPCCTR$_4$.ISM=11
iom_mon_2_i(5)	IOM_MON2_5	CAN node 0 output: TXDCAN0	IOM_FPCCTR$_5$.ISM=11
iom_mon_2_i(6)	IOM_MON2_6	CAN node 1 output: TXDCAN1	IOM_FPCCTR$_6$.ISM=11
iom_mon_2_i(7)	IOM_MON2_7	CAN node 2 output: TXDCAN2	IOM_FPCCTR$_7$.ISM=11
iom_mon_2_i(8)	IOM_MON2_8	GTM output: TOUT64	IOM_FPCCTR$_8$.ISM=11
iom_mon_2_i(9)	IOM_MON2_9	GTM output: TOUT104	IOM_FPCCTR$_9$.ISM=11
iom_mon_2_i(10)	IOM_MON2_10	GTM output: TOUT105	IOM_FPCCTR$_{10}$.ISM=11
iom_mon_2_i(11)	IOM_MON2_11	GTM output: TOUT65	IOM_FPCCTR$_{11}$.ISM=11
iom_mon_2_i(12)	IOM_MON2_12	ASCLIN0 output: ATX0	IOM_FPCCTR$_{12}$.ISM=11
iom_mon_2_i(13)	IOM_MON2_13	ASCLIN1 output: ATX1	IOM_FPCCTR$_{13}$.ISM=11
iom_mon_2_i(14)	IOM_MON2_14	GTM output: TOUT66	IOM_FPCCTR$_{14}$.ISM=11
iom_mon_2_i(15)	IOM_MON2_15	GTM output: TOUT67	IOM_FPCCTR$_{15}$.ISM=11

Figure 28-13 Monitor portlogic/block signal connectivity

IOM Signal Name	Pinning Table Signal	Description	Selection setting
iom_pin_i(0)	IOM_PIN0	GPIO pad input	IOM_FPCCTR$_0$.ISR=000
iom_pin_i(1)	IOM_PIN1	GPIO pad input	IOM_FPCCTR$_1$.ISR=000
iom_pin_i(2)	IOM_PIN2	GPIO pad input	IOM_FPCCTR$_2$.ISR=000
iom_pin_i(3)	IOM_PIN3	GPIO pad input	IOM_FPCCTR$_3$.ISR=000
iom_pin_i(4)	IOM_PIN4	GPIO pad input	IOM_FPCCTR$_4$.ISR=000
iom_pin_i(5)	IOM_PIN5	GPIO pad input	IOM_FPCCTR$_5$.ISR=000
iom_pin_i(6)	IOM_PIN6	GPIO pad input	IOM_FPCCTR$_6$.ISR=000
iom_pin_i(7)	IOM_PIN7	GPIO pad input	IOM_FPCCTR$_7$.ISR=000
iom_pin_i(8)	IOM_PIN8	GPIO pad input	IOM_FPCCTR$_8$.ISR=000
iom_pin_i(9)	IOM_PIN9	GPIO pad input	IOM_FPCCTR$_9$.ISR=000
iom_pin_i(10)	IOM_PIN10	GPIO pad input	IOM_FPCCTR$_{10}$.ISR=000
iom_pin_i(11)	IOM_PIN11	GPIO pad input	IOM_FPCCTR$_{11}$.ISR=000
iom_pin_i(12)	IOM_PIN12	GPIO pad input	IOM_FPCCTR$_{12}$.ISR=000
iom_pin_i(13)	IOM_PIN13	GPIO pad input	IOM_FPCCTR$_{13}$.ISR=000
iom_pin_i(14)	IOM_PIN14	GPIO pad input	IOM_FPCCTR$_{14}$.ISR=000
iom_pin_i(15)	IOM_PIN15	GPIO pad input	IOM_FPCCTR$_{15}$.ISR=000
iom_ref_0_i(0)	IOM_REF0_0	GTM output : TOUT0	IOM_FPCCTR$_0$.ISR=001
iom_ref_0_i(1)	IOM_REF0_1	GTM output : TOUT1	IOM_FPCCTR$_1$.ISR=001
iom_ref_0_i(2)	IOM_REF0_2	GTM output : TOUT2	IOM_FPCCTR$_2$.ISR=001
iom_ref_0_i(3)	IOM_REF0_3	GTM output : TOUT3	IOM_FPCCTR$_3$.ISR=001
iom_ref_0_i(4)	IOM_REF0_4	GTM output : TOUT4	IOM_FPCCTR$_4$.ISR=001
iom_ref_0_i(5)	IOM_REF0_5	GTM output : TOUT5	IOM_FPCCTR$_5$.ISR=001
iom_ref_0_i(6)	IOM_REF0_6	GTM output : TOUT6	IOM_FPCCTR$_6$.ISR=001
iom_ref_0_i(7)	IOM_REF0_7	GTM output : TOUT7	IOM_FPCCTR$_7$.ISR=001
iom_ref_0_i(8)	IOM_REF0_8	GTM output : TOUT8	IOM_FPCCTR$_8$.ISR=001
iom_ref_0_i(9)	IOM_REF0_9	GTM output : TOUT9	IOM_FPCCTR$_9$.ISR=001
iom_ref_0_i(10)	IOM_REF0_10	GTM output : TOUT10	IOM_FPCCTR$_{10}$.ISR=001
iom_ref_0_i(11)	IOM_REF0_11	GTM output : TOUT11	IOM_FPCCTR$_{11}$.ISR=001
iom_ref_0_i(12)	IOM_REF0_12	GTM output : TOUT12	IOM_FPCCTR$_{12}$.ISR=001
iom_ref_0_i(13)	IOM_REF0_13	GTM output : TOUT13	IOM_FPCCTR$_{13}$.ISR=001
iom_ref_0_i(14)	IOM_REF0_14	GTM output : TOUT14	IOM_FPCCTR$_{14}$.ISR=001
iom_ref_0_i(15)	IOM_REF0_15	GTM output : TOUT15	IOM_FPCCTR$_{15}$.ISR=001
iom_ref_1_i(0)	IOM_REF1_0	CCU60 output : COUT63	IOM_FPCCTR$_0$.ISR=010
iom_ref_1_i(1)	IOM_REF1_1	CCU60 output : COUT62	IOM_FPCCTR$_1$.ISR=010
iom_ref_1_i(2)	IOM_REF1_2	CCU60 output : COUT61	IOM_FPCCTR$_2$.ISR=010
iom_ref_1_i(3)	IOM_REF1_3	CCU60 output : COUT60	IOM_FPCCTR$_3$.ISR=010
iom_ref_1_i(4)	IOM_REF1_4	CCU60 output : CC62	IOM_FPCCTR$_4$.ISR=010
iom_ref_1_i(5)	IOM_REF1_5	CCU60 output : CC61	IOM_FPCCTR$_5$.ISR=010
iom_ref_1_i(6)	IOM_REF1_6	CCU60 output : CC60	IOM_FPCCTR$_6$.ISR=010
iom_ref_1_i(7)	IOM_REF1_7	CCU61 output : COUT63	IOM_FPCCTR$_7$.ISR=010
iom_ref_1_i(8)	IOM_REF1_8	CCU61 output : COUT62	IOM_FPCCTR$_8$.ISR=010
iom_ref_1_i(9)	IOM_REF1_9	CCU61 output : COUT61	IOM_FPCCTR$_9$.ISR=010
iom_ref_1_i(10)	IOM_REF1_10	CCU61 output : COUT60	IOM_FPCCTR$_{10}$.ISR=010
iom_ref_1_i(11)	IOM_REF1_11	CCU61 output : CC62	IOM_FPCCTR$_{11}$.ISR=010
iom_ref_1_i(12)	IOM_REF1_12	CCU61 output : CC61	IOM_FPCCTR$_{12}$.ISR=010
iom_ref_1_i(13)	IOM_REF1_13	CCU61 output : CC60	IOM_FPCCTR$_{13}$.ISR=010
iom_ref_1_i(14)	IOM_REF1_14	GTM output : TOUT81	IOM_FPCCTR$_{14}$.ISR=010
iom_ref_1_i(15)	IOM_REF1_15	GTM output : TOUT82	IOM_FPCCTR$_{15}$.ISR=010
iom_ref_2_i(0)	IOM_REF2_0	QSPI0 output : MRST0	IOM_FPCCTR$_0$.ISR=011
iom_ref_2_i(1)	IOM_REF2_1	QSPI1 output : MRST1	IOM_FPCCTR$_1$.ISR=011
iom_ref_2_i(2)	IOM_REF2_2	QSPI2 output : MRST2	IOM_FPCCTR$_2$.ISR=011
iom_ref_2_i(3)	IOM_REF2_3	GQSI3 output : MRST3	IOM_FPCCTR$_3$.ISR=011
iom_ref_2_i(4)	IOM_REF2_4	GTM output : TOUT83	IOM_FPCCTR$_4$.ISR=011
iom_ref_2_i(5)	IOM_REF2_5	CAN node 0 output : TXDCAN0	IOM_FPCCTR$_5$.ISR=011
iom_ref_2_i(6)	IOM_REF2_6	CAN node 1 output : TXDCAN1	IOM_FPCCTR$_6$.ISR=011
iom_ref_2_i(7)	IOM_REF2_7	CAN node 2 output : TXDCAN2	IOM_FPCCTR$_7$.ISR=011
iom_ref_2_i(8)	IOM_REF2_8	GTM output : TOUT84	IOM_FPCCTR$_8$.ISR=011
iom_ref_2_i(9)	IOM_REF2_9	GTM output : TOUT107	IOM_FPCCTR$_9$.ISR=011
iom_ref_2_i(10)	IOM_REF2_10	GTM output : TOUT108	IOM_FPCCTR$_{10}$.ISR=011
iom_ref_2_i(11)	IOM_REF2_11	GTM output : TOUT85	IOM_FPCCTR$_{11}$.ISR=011
iom_ref_2_i(12)	IOM_REF2_12	ASCLIN0 output : ATX0	IOM_FPCCTR$_{12}$.ISR=011
iom_ref_2_i(13)	IOM_REF2_13	ASCLIN1 output : ATX1	IOM_FPCCTR$_{13}$.ISR=011
iom_ref_2_i(14)	IOM_REF2_14	GTM output : TOUT86	IOM_FPCCTR$_{14}$.ISR=011
iom_ref_2_i(15)	IOM_REF2_15	GTM output : TOUT87	IOM_FPCCTR$_{15}$.ISR=011
-	- -	EXOR Combiner o/p	IOM_FPCCTR$_k$.ISR=1xx where k = channel number (0 to 15)
			IOM_PORTLOGIC_CONN_REF_EP

Figure 28-14 Reference portlogic/block signal connectivity

IOM Signal Name	Pinning Table Signal	Description	Inclusion setting
iom_gtm_i(0)	-	GTM output : TOUT22	IOM_GTMEXR.EN0=1 (set to 0 to exclude)
iom_gtm_i(1)	-	GTM output : TOUT23	IOM_GTMEXR.EN1=1 (set to 0 to exclude)
iom_gtm_i(2)	-	GTM output : TOUT24	IOM_GTMEXR.EN2=1 (set to 0 to exclude)
iom_gtm_i(3)	-	GTM output : TOUT25	IOM_GTMEXR.EN3=1 (set to 0 to exclude)
iom_gtm_i(4)	-	GTM output : TOUT26	IOM_GTMEXR.EN4=1 (set to 0 to exclude)
iom_gtm_i(5)	-	GTM output : TOUT27	IOM_GTMEXR.EN5=1 (set to 0 to exclude)
iom_gtm_i(6)	-	GTM output : TOUT28	IOM_GTMEXR.EN6=1 (set to 0 to exclude)
iom_gtm_i(7)	-	GTM output : TOUT29	IOM_GTMEXR.EN7=1 (set to 0 to exclude)

IOM_PORTLOGIC_CONN_EXOR_EP

Figure 28-15 EXOR combiner portlogic/block signal connectivity

28.11 Example Monitor/Safety Measures

For each channel the following measurements shall be possible:

- Duty cycle measurement
- Set-up and/or hold time measurement
- Delay window (min,max) with respect to rising or falling edge

The following waveforms provide more examples of typical signal monitoring functionality that could be deployed using the IOM. Some typical configuration settings are also provided on some of the examples.

28.11.1 Example 1 - Pulse or duty cycle too short

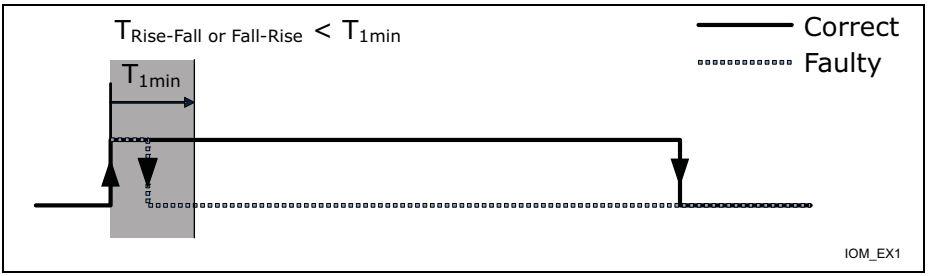


Figure 28-16 example 1 - pulse or duty cycle too short

Example settings for LAM block registers:

LAMCFG.IVR: 0x0 ; don't care

LAMCFG.IVM: 0x0 ; don't invert monitor signal (or invert for non-duty part of period)

LAMCFG.MOS: 0x0 ; monitor signal used for event generation

LAMCFG.RMS: 0x0 ; free-running

LAMCFG.EWS: 0x1 ; select monitor signal for event window

LAMCFG.EDS: 0x9 ; CLR on positive edge, EVT on negative edge

LAMCFG.IVW : 0x1 ; invert window, capture events under the threshold

LAMCFG.MCS/RCS ; select according to device configuration

LAMEWS.THR ; select appropriate threshold (minimum duty cycle length required. If duty cycle is shorter than this value then EVT will trigger)

28.11.2 Example 2 - Pulse or duty cycle too long

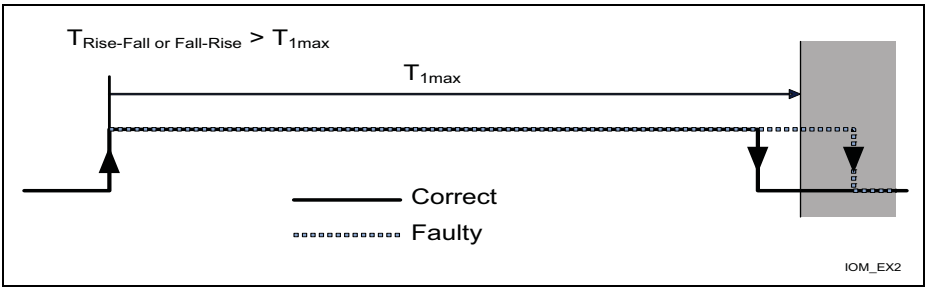


Figure 28-17 example 2 - pulse or duty cycle too long

Example settings for LAM block registers:

LAMCFG.IVR: 0x0 ; don't care

LAMCFG.IVM: 0x0 ; don't invert monitor signal (or invert for non-duty part of period)

LAMCFG.MOS: 0x0 ; monitor signal used for event generation

LAMCFG.RMS: 0x0 ; free-running

LAMCFG.EWS: 0x1 ; select monitor signal for event window

LAMCFG.EDS: 0x9 ; CLR on positive edge, EVT on negative edge

LAMCFG.IVW : 0x0 ; don't invert window, capture events beyond the threshold

LAMCFG.MCS/RCS ; select according to device configuration

LAMEWS.THR ; select appropriate threshold (maximum duty cycle length required. If duty cycle is longer than this value then EVT will trigger)

28.11.3 Example 3 - Period too short

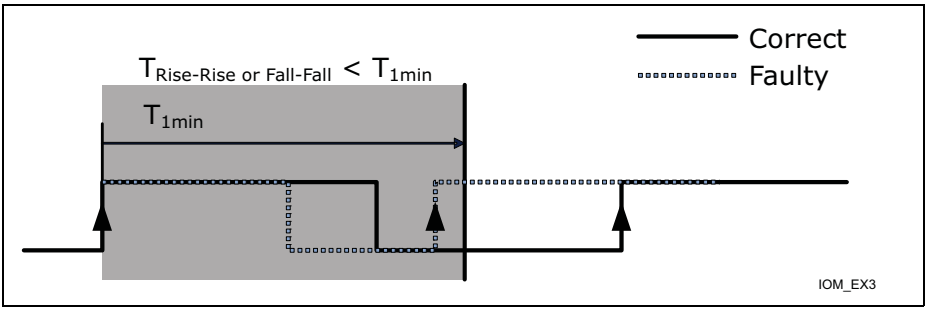


Figure 28-18 example 3 - period too short

Example settings for LAM block registers:

LAMCFG.IVR: 0x0 ; don't care

LAMCFG.IVM: 0x0 ; don't invert monitor signal

LAMCFG.MOS: 0x0 ; monitor signal used for event generation

LAMCFG.RMS: 0x0 ; free-running

LAMCFG.EWS: 0x1 ; select monitor signal for event window

LAMCFG.EDS: 0x5 ; CLR on positive edge, EVT on positive edge

LAMCFG.IVW : 0x1 ; invert window, capture events under the threshold

LAMCFG.MCS/RCS ; select according to device configuration

LAMEWS.THR ; select appropriate threshold (minimum period length required. If duty cycle is shorter than this value then EVT will trigger)

28.11.4 Example 4 - Period too long

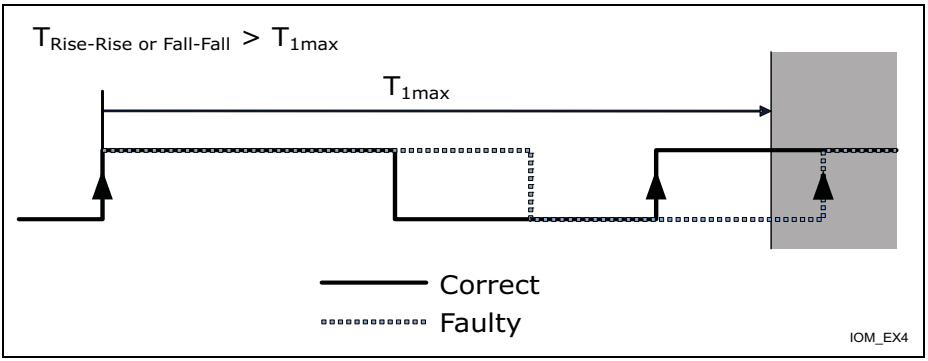


Figure 28-19 example 4 - period too long

Example settings for LAM block registers:

LAMCFG.IVR: 0x0 ; don't care

LAMCFG.IVM: 0x0 ; don't invert monitor signal

LAMCFG.MOS: 0x0 ; monitor signal used for event generation

LAMCFG.RMS: 0x0 ; free-running

LAMCFG.EWS: 0x1 ; select monitor signal for event window

LAMCFG.EDS: 0x5 ; CLR on positive edge, EVT on positive edge

LAMCFG.IVW : 0x0 ; don't invert window, capture events beyond the threshold

LAMCFG.MCS/RCS ; select according to device configuration

LAMEWS.THR ; select appropriate threshold (maximum period length required. If duty cycle is longer than this value then EVT will trigger)

28.11.5 Example 5 - Diagnosis of Command and Feedback - acceptable propagation window and/or signal consistency check

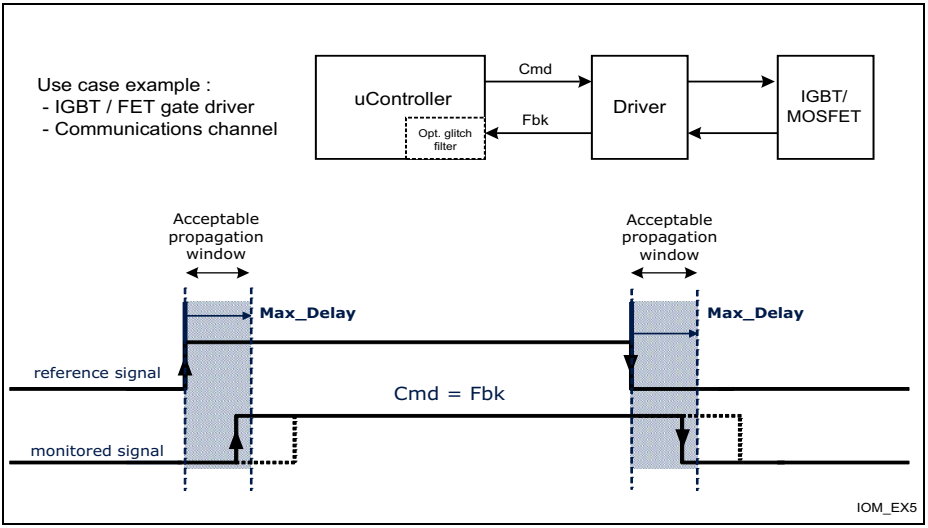


Figure 28-20 example 5 - Diagnosis of Command & Feedback

Using the driven signal (off-chip) as a reference, using a feedback signal (input to device) as the monitor. Checking that the monitored signal shows activity within an acceptable propagation window, and/or that the feedback signal shows the same behaviour as the reference (Command = Feedback).

Example settings for LAM block registers:

LAMCFG.IVR: 0x0 ; don't invert reference signal

LAMCFG.IVM: 0x0 ; don't invert monitor signal

LAMCFG.MOS: 0x1 ; (MON xor REF) signal used for event generation

LAMCFG.RMS: 0x0 ; free-running

LAMCFG.EWS: 0x0 ; select reference signal for event window

LAMCFG.EDS: 0xB ; CLR on both edges of REF, EVT on falling edge (of xor'd signal)

LAMCFG.IVW : 0x0 ; don't invert window, capture events beyond the threshold

LAMCFG.MCS/RCS ; select according to device configuration

LAMEWS.THR ; set to max delay allowed

28.11.6 Example 6 - Diagnosis of Set-up and Hold times

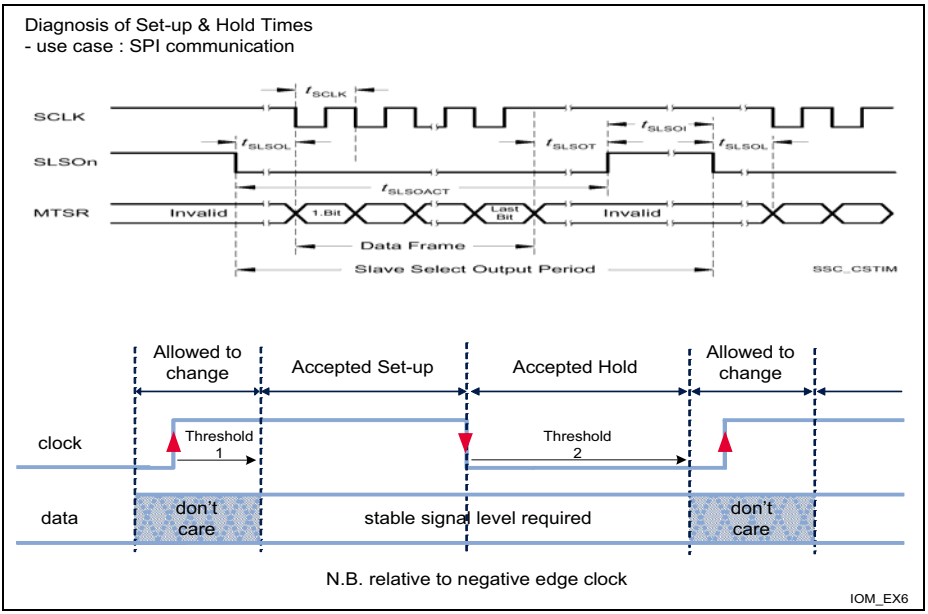


Figure 28-21 example 6 - Diagnosis of Set-up & Hold times

Ensuring that data remains stable within the Set-up and Hold windows, generating an event otherwise, and/or ensuring that data only changes within the specified windows.

Note that 2 LAM channels will be required for this purpose, one for set-up, one for hold.

Example settings for LAM block registers for Set-up :

LAMCFG.IVR: 0x0 ; don't invert reference signal

LAMCFG.IVM: 0x0 ; don't invert monitor signal

LAMCFG.MOS: 0x0 ; monitor signal used for event generation

LAMCFG.RMS: 0x1 ; run is gated

LAMCFG.EWS: 0x0 ; select reference signal for event window

LAMCFG.EDS: 0xF ; CLR on both edges, EVT on both edges

LAMCFG.IVW : 0x0 ; don't invert window, capture events beyond the threshold

LAMCFG.MCS/RCS ; select according to device configuration

LAMEWS.THR ; Acceptable Set-up (ref Threshold 1 on waveforms shown)

Example settings for LAM block registers for Hold:

LAMCFG.IVR: 0x0 ; invert reference signal (use for gating)

LAMCFG.IVM: 0x0 ; don’t invert monitor signal

LAMCFG.MOS: 0x0 ; monitor signal used for event generation

LAMCFG.RMS: 0x1 ; event window is gated

LAMCFG.EWS: 0x0 ; select reference signal for event window

LAMCFG.EDS: 0xD ; CLR on positive edge (inverted neg. edge), EVT on both edges

LAMCFG.IVW : 0x1 ; invert window, capture events below the threshold

LAMCFG.MCS/RCS ; select according to device configuration

LAMEWS.THR ; Acceptable Hold (ref Threshold 2 on waveforms shown)

29 Advanced Driver Assistance Subsystem (ADAS)

Some product derivatives include hardware support for Advanced Driver Assistance Systems (See Introduction chapter).

In ADAS derivative products the following additional hardware is present:

- Camera and RADAR Interface (CIF) (TC26x and TC29x only)
  - Parallel interface
  - On-the-fly JPEG encoding of camera signals
  - De-interleaving of external ADC radar signals
- Fast Fourier Transform Accelerator (FFT)
  Integrated windowing support and up to 2048 point FFTs
- Extended Memory
- A high speed Backbone Bus
- Additional ADC channels (TC23x only)
  - Supports 4 channel concurrent sampling throughout all ADAS products
- A high speed input capture unit
  - For high precision frequency measurements of low speed signals
  - See QSPI module, section “High Speed Input Capture”

This chapter describes these peripherals in the context of radar and video based applications for Advanced Driver Assistance Systems

Figure 29-1 shows an overview of the ADAS subsytem.

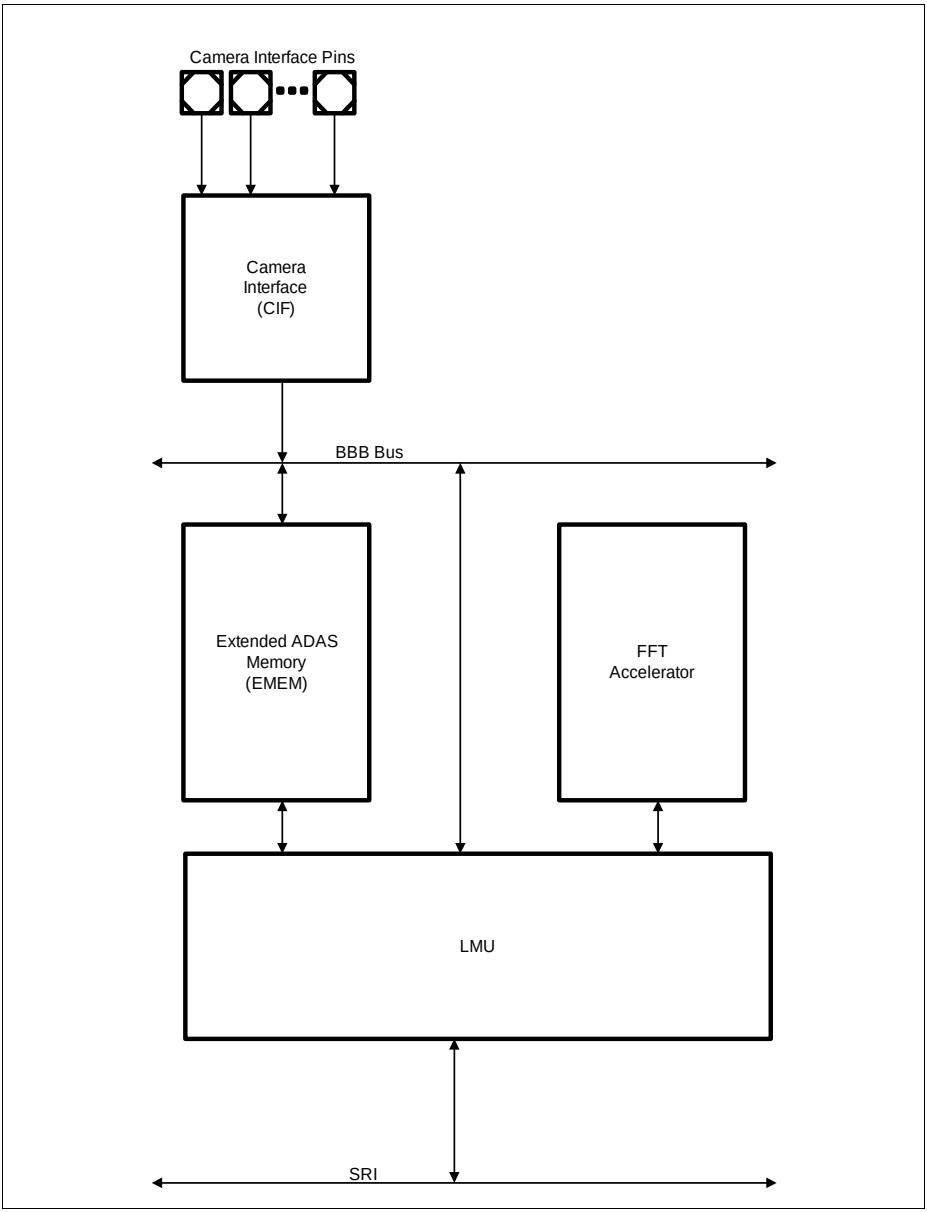


Figure 29-1 Block Diagram of the ADAS Subsystem

29.1 FFT Engine (FFT)

This chapter describes the FFT Engine, an algorithm specific accelerator. The FFT Engine is not available in all devices and only those with the ADAS feature set.

The FFT Engine utilises the SRI connectivity of the LMU. The FFT Engine is clocked at the same frequency as the SRI and all cycle counts are in terms of SRI clocks.

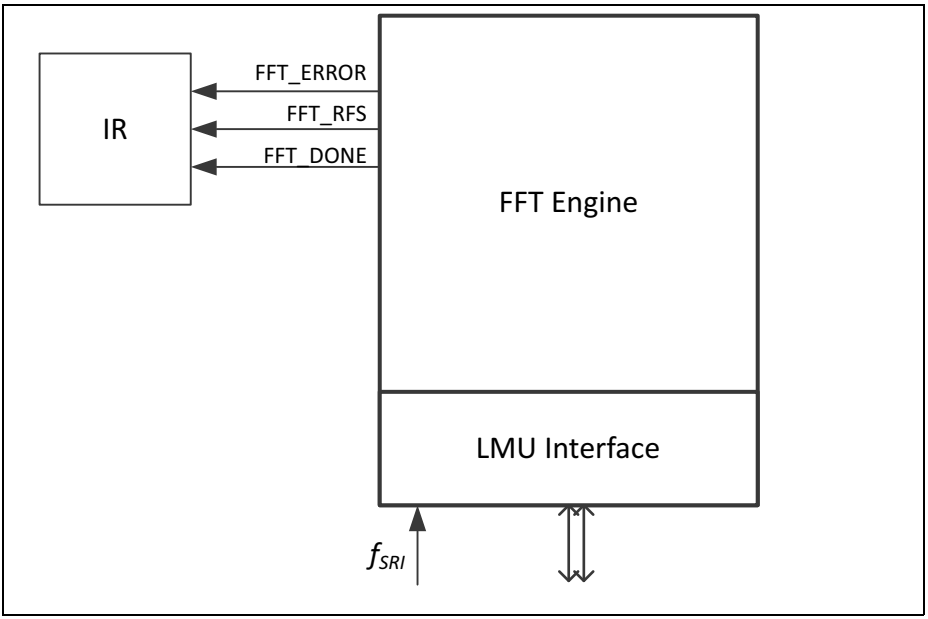


Figure 29-2 FFT Engine Integration into the LMU

29.1.1 Overview

The FFT Engine calculates forward and inverse FFTs of lengths 8 to 2048 (powers of 2 only). It operates on real and complex data which can have co-efficients of 16 or 32-bits in length.

The actual forward and inverse transforms calculated by this core are given by

$$X[k] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]\exp(-j2\pi kn/N) \qquad \text{Forward}$$

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]\exp(j2\pi kn/N) \qquad \text{Inverse}$$

In usual DSP literature the forward transform is not divided by N, but this is performed in hardware for consistency between transforms and to implement a transform that cannot overflow.

29.1.2 Feature List

General features:

- Per transform selection of FFT or inverse FFT (IFFT).
- Per transform selection of FFT/IFFT size from 8, 16, 32, 64, 128, 256, 512, 1024 or 2048 points.
- Per transform selection of real or complex input operands.
- Per transform selection of input operands with 16-bit or 32-bit precision.
- Per transform selection of output results with 16-bit or 32-bit precision.
- Internal calculation to 32-bits of precision.

In addition to standard FFT calculations, the FFT Engine also supports a window function to be applied to input operands. The window function utilises a window RAM to hold coefficients which allows any real symmetrical function to be used. Use of this features is described in **Section 29.1.7**. Another important performance enhancement is the use of double buffering to support the overlap of loading and unloading the FFT Engine while computation continues

Input operands for FFT or inverse FFT calculations which are loaded with 16-bits of precision are converted to 32-bits prior to storage inside the FFT engine. Calculations are carried out to 32-bits of precision using block floating point and adaptive scaling to ensure there is no loss of dynamic range during the intermediate calculations. Results are adjusted to natural scaling for software convenience before being read out. Output results which are unloaded with 16-bits of precision are converted from the 32-bit internal result by rounding (to nearest even) with saturation as they are transferred out.

29.1.3 FFT Engine Performance

After operands are loaded into the FFT Engine, the number of cycles the engine takes to complete a computation is a function of the length of the transform it has been programmed to execute. If N is the length of the transform, then the number of clock cycles to complete is:

```
(N+16)*ceil(log4(N))
```

The cycle count till availability of results for the supported FFT/IFFT lengths is show in **Table 29-1**.

Table 29-1 FFT Engine Execution Times

Transform Length	Cycles	Transform Length	Cycles
-	-	128	576
8	48	256	1088
16	64	512	2640
32	144	1024	5200
64	240	2048	12384

29.1.4 System Performance

The complete steps required for the execution of an FFT/IFFT are:

- writing to the FFT_CSR to configure the required operation (input format, length, window operation, output format, start) with a single write
- transferring the required number of operands into the FFT Engine
- waiting for the engine to complete (or being interrupted)
- unloading the results from the FFT Engine

Total execution time for an FFT/IFFT calculation must take into consideration the time for storing of operands into the FFT Engine, and reading of the results out as well as the calculation time shown in the table FFT Engine Execution Times.

29.1.4.1 Data Movement Optimisation

The loading and unloading cycles will be a significant part of the total execution time required and is one of the major optimisation opportunities. Using the minimum operand size for input or output (16-bit versus 32-bit) can halve the amount of time in loading and unloading.

FFT transforms based on real operands only, could be calculated by loading complex values with imaginary components of zero along with real component values. However the FFT Engine supports optimising out this unnecessary transfer, and setting FFT_CSR.IN_FMT appropriatly for real operands will halve the loading time. Both these transform options will significantly reduce the end to end cycle count for an operation, despite the fact that they have no effect on the transform calculation time.

Programmed loading and unloading by TriCore CPUs can transfer 64-bits at a time. For smaller transform lengths this may be most efficient. For longer transforms, the optimum bandwidth to load and unload the FFT Engine is obtained using the DMA to transfer bursts of four 64-bit words at a time.

29.1.4.2 Double Buffering Optimisation

As well as the opportunity to optimise the data movement time, the FFT Engine also provide the mechanisms to overlap FFT/IFFT execution with loading and unloading results. The FFT Engine is provided with two buffers each capable of supporting a maximum length FFT/IFFT transform. These two buffers are operated in a ping-pong arrangement and allow the overlap of transfer in and out of the FFT Engine, with the calculation of a transform with an earlier loaded set of operands. Thus, the throughput of FFT/IFFT operations is higher than determined by the total latency (loading, execution, unloading) of an individual transform.

29.1.5 Address Spaces and Access Protection

The FFT Engine responds to three address spaces in the system address map:

- a register space located from F870_0C00h to F870_0CFFh,
- a data space located at BE00_0000h to BE07_FFFFh,
- a window coefficient space located at BE10_0000h to BE17_FFFFh.

All three addresses spaces within the FFT Engine are protected by the access enable registers of the LMU which protects the resources against writes from unauthorised SoC masters.

29.1.5.1 Register Address Space

The FFT register space consists of 32-bit registers that only respond correctly to 32-bit read and write operations.

- any read from register space which is an 8-,16- or 64-bit operation will result in an undefined value,
- any read from register space which is a burst transfer will result in a transaction error,
- any write to register space which is a burst transfer or an 8-,16- or 64-bit operation will result in a transaction error,
- any atomic operation to register space will result in a transaction error.

29.1.5.2 Data Space

The data space only supports 64-bit and burst-4 transfers. Accesses that are 8-,16- or 32-bit or burst-2 transfers to the data space will have undefined results.

The data space does not have standard memory characteristics. Instead this region operates as access to two separate FIFOs. Write accesses (regardless of addresses used) will place operands in order into one of the ping-pong buffers; for example for a 512 length FFT, the first data transferred will be ‘sample 0’, and the 512th sample transferred will be ‘sample 511’ regardless of the addresses used (provided they are all

within the FFT Engine data space). There is no ability to direct accesses to either of the ping-pong buffers, this is purely controlled by the FFT Engine.

Read accesses to the data space will return results out of one of the FIFOs in the same manner. The first read of a completed transform will receive ‘result 0’, and the Nth read will receive ‘result N-1’, again the addresses used will not influence the result element returned.

Bit FFTPFDIS should be cleared (0) in register LMU_MEMCON (see Local Memory Unit chapter) before accessing the data space.

Intermixing single transfers of 64-bits or burst-4 transfers during the same loading phase, or same unloading phase will have undefined results.

29.1.5.3 Window Coefficient Space

The widow coefficient space only supports 64-bit and burst-4 transfers. Accesses that are 8-,16- or 32-bit or burst-2 transfers to the window space will have undefined results. Within the window coeffcient address space is a RAM holding the window coefficients - the window RAM.

The window RAM occupies 2-KByte (with corresponding aliases) of the window address space. Read and writes to this 2-KByte RAM will have standard memory characteristics only when validly accessed.

Accesses to the window RAM are only well behaved (reads to an address will return the same data last written) when

- the FFT Engine is idle,
- the FFT Engine does not have any results to be unloaded and
- bit FFTPFDIS should be set (1) in register LMU_MEMCON (see Local Memory Unit chapter).

For this reason writing to the window RAM should only be performed when the three conditions above have been met.

Note: The entire window RAM must be initialised before the FFT Engine can run a transform. This requirement is regardless of whether the window function is enabled or not for the transform.

29.1.6 Data Value Formats

The data formats supported by the FFT Engine for operands are identical to the corresponding TriCore architecture fractional formats for ease of manipulation. While, the window coefficient format which is manipulated less frequently uses a different format to slightly enhance precision.

29.1.6.1 Time and Frequency Sample Format

Data with 16-bit precision corresponds to TriCore signed fractional data with 16-bits (also known as 1.15 or Q15 format). Data with 32-bit precision corresponds to TriCore signed fractional data with 32-bits (also known as 1.31 or Q31 format).

Data values in this format have a single high-order sign bit, where 0 represents positive (+) and 1 represents negative (-), followed by an implied binary point and fraction. Their values are therefore in the range [-1,1).

Complex data corresponds to an array of two elements in little-endian order, where the first element (index 0 in ‘C’ notation) is the real coefficient, and the second element (index 1 in ‘C’ notation) is the imaginary coefficient.

29.1.6.2 Window Coefficient Format

A valid window function will have coefficients in the range [0,1]. However, strictly only the central window coefficient requires to be 1.0 and the remainder are in the range [0,1). If the 1.15 or Q15 format was used for window coefficients, then as all the coefficients are positive, the MSBs of co-efficients stored in the window RAM (the central coefficient is not loaded), would be redundant.

To take advantage of this fact, the FFT Engine calculate with unsigned window coefficients and the stored values to the window RAM are the 16 fractional bits of a nominal 1.16 format.

29.1.7 Window Usage

The window RAM allows the user to load a window function of their choice and apply it (or not) to subsequent transforms. The parameters in the window RAM are only changed when the window function is required to be changed, they otherwise persist between transform operations (i.e. have standard memory characteristics).

The window RAM only contains half the maximum length transform entries (see **Section 29.1.6.2** for the format) and exploits symmetry to generate a complete window for that length of transform. If a window function is to be used for any length transform, it must be fully loaded and the FFT Engine will select the appropriate coefficients.

The expected usage is that a window function will be selected for a good match to the system and application properties, loaded at system initialisation, and then either applied or not (using FFT_CSR.WIN_BYP clear or set respectively) to subsequent FFT operations. While the window function can be changed during system operation, any change must respect the requirements listed in **Section 29.1.5.3**. Thus, any change will incur some latency till the next transform operation is possible; either with or without the new window function applied.

*Note: As specified in **Section 29.1.5.3** the window RAM must be initialised before any FFT Engine transform operation.*

29.1.7.1 Window Parameters

If the window function is to be used, the coefficients for a maximum length transform must be computed. First generate a window of odd length and with a central maximum value of 1.0. For example if using MATLAB then this command

```
w = hamming(2048,'periodic');
```

or

```
w = hamming(2049,'symmetric');
```

will return the first 2048 points of a 2049 point periodic window. Next select the first 1024 values of w and scale them up and round to an unsigned 16-bit integer

```
wi = round(65536*w(1:1024));
```

These are the values to load into the window RAM. Note that this procedure is generic for any window, always start with a length 2049 symmetric or 2048 period variant. A common window in radar is Blackman-Harris, which would be generated in MATLAB with

```
w = blackmanharris(2049);
wi = round(65536*w(1:1024));
```

Note that the 1.0 central coefficient is internally generated by the hardware so it is essential that the window is correctly scaled to 1.0.

For FFT transform sizes less than the maximum the hardware will index the full window table to generate the correct coefficients for a smaller FFT size.

Although the hardware will not stop you applying a window in the frequency domain it may not be desirable.

29.1.8 Interrupt and DMA Operation

As well as inspecting the CSR register to determine when FFT/IFFT operations may be commenced, the FFT Engine allows interrupt driven data movement. There are two SRC registers in the IR connected to triggers from the FFT Engine. These are SRC_FFTRFS and SRC_FFTDONE. As is standard, each of these SRCs can select either a CPU or DMA channel to service the event.

A trigger will be sent to SRC_FFTRFS whenever FFT_CSR.RFS transitions from 0 to 1, this alerts the system to the fact the FFT_ENGINE can be loaded with another transform. Since there are two (ping-pong) buffers, this event means the FFT Engine may still be busy computing a previously loaded transform.

A trigger will be sent to SRC_FFTDONE whenever a transform calculation has completed and the unload operation can be performed.

A possible operational sequence utilising both of these trigger conditions is described below

1. A CPU reads the FFT_CSR register and looks for RFS bit to be set. Software then writes to the FFT_CSR register with START set to 1, and the desired values for IFFT, LENGTH, WIN_BYP, IN_FMT, OUT_FMT.
2. The same CPU programs DMA channel(s) to send data to the FFT Engine, and configures the service request control (SRC) registers in the interrupt router SRC_FFTRFS and SRC_FFTDONE as required.
   At this stage the CPU could return to some other task (perhaps preparing operands or examining results).
3. A CPU receiving the interrupt corresponding to SRC_FFTRFS, can write the next START command and initiate the DMA to transfer the next transforms operands to the FFT Engine.
   Again at this stage the CPU is then free to return to some other task.
4. An interrupt corresponding to SRC_FFTDONE can trigger an already configured DMA channel to read back data from the FFT Engine, which when it has unloaded the results can then trigger a CPU to inform it of the unload completion.
5. Loop back to 3

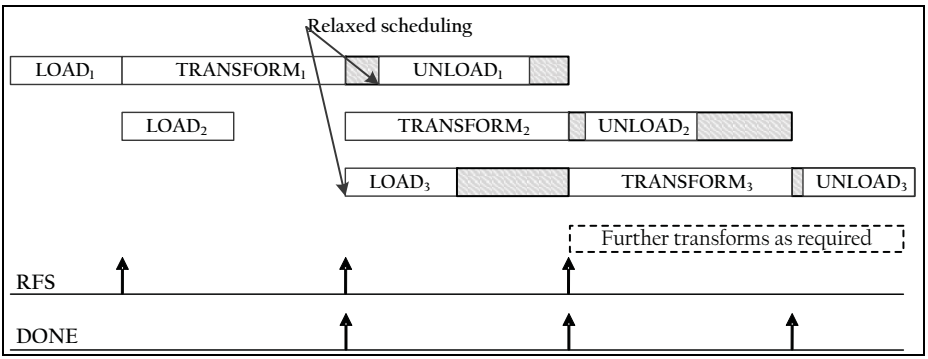


Figure 29-3 Three Consecutive Transforms

The double buffer architecture allows overlapping FFT execution with the DMA load/unload operation. This is illustrated in **Figure 29-3** where this example shows three transforms. The use of two memory buffers allows more relaxed scheduling of the DMA load/unload during overlapped transforms as shown by the unload operation being delayed until after the load operation for the third transform has commenced.

29.1.9 FFT Engine Registers

This section described the register interface to the FFT Engine. This address region should not be unintentionally read or written because this might cause side effects on FFT Engine state.

All registers are reset by the application reset apart from **FFT_ODA** and **FFT_OCS**. These registers are cleared by Debug Reset and by each System Reset when OCDS is disabled. They are not changed by System Reset when OCDS is enabled.

Where a register is defined as "ACCEN Enable Protected" (designated with "P" for writes), then the LMU ACCEN settings will be used.

Table 29-2 Registers Address Space

Module	Base Address	End Address	Note
FFT	F870 0C00$_H$	F870 0CFF$_H$	-

Table 29-3 Registers Overview

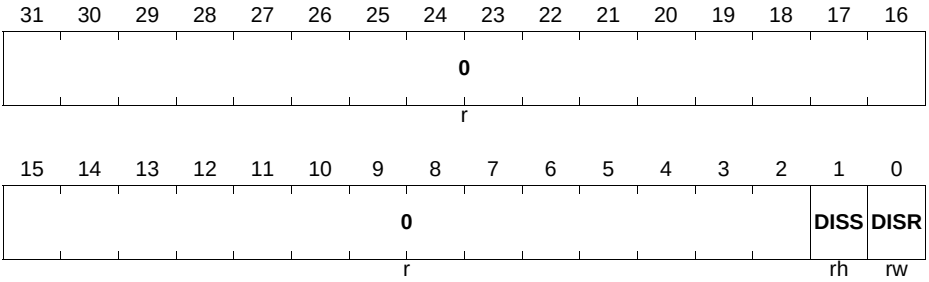
Register Short Name	Register Long Name	Offset Address	Read Access	Write Access	Reset Value
FFT_CLC	FFT Clock Control Register	00$_H$	U/SV	SV,E,P	0000 0002$_H$
FFT_ID	FFT Identification Register	08$_H$	U/SV	R	00E2 C002$_H$
FFT_CSR	FFT Control and Status Register	40$_H$	U/SV	U/SV,P	0010 0000$_H$
FFT_HISTORY0	FFT History Register 0	60$_H$	U/SV	R	0000 0000$_H$
FFT_HISTORY1	FFT History Register 1	70$_H$	U/SV	R	0000 0000$_H$
FFT_ODA	FFT OCDS Debug Access Register	E4$_H$	U/SV	SV,P	0000 0000$_H$
FFT_OCS	FFT OCDS Control and Status Register	E8$_H$	U/SV	SV,P	0000 0000$_H$
FFT_KRSTCLR	FFT Kernel Reset Status Clear Register	EC$_H$	U/SV	SV,P	0000 0000$_H$
FFT_KRST1	FFT Kernel Reset Register 1	F0$_H$	U/SV	SV, P	0000 0000$_H$
FFT_KRST0	FFT Kernel Reset Register 0	F4$_H$	U/SV	SV, P	0000 0000$_H$

29.1.9.1 FFT Clock Control Register

FFT Clock Control Register (FFT_CLC)

The FFT Clock Control Register, FFT_CLC, acts globally and allows the complete FFT Engine to be disabled to reduce power consumption when the FFT is not required. When the FFT Engine is disabled (DISS=1_B), only register accesses to the FFT Engine are permitted. All other accesses to the FFT address space will receive a transaction error.

FFT_CLC
FFT Clock Control Register (0000_H) Reset Value: $0000\ 0002_H$



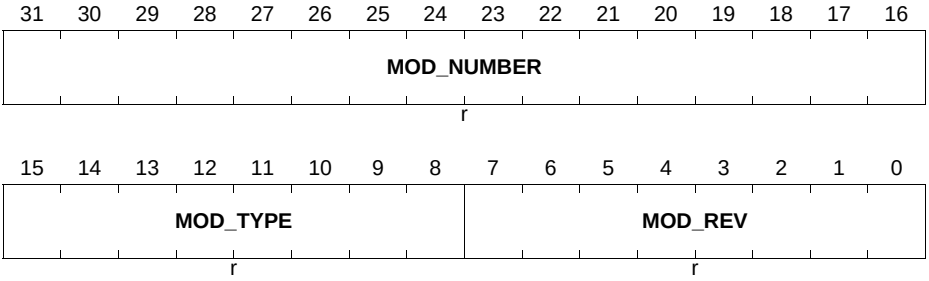
Field	Bits	Type	Description
DISR	0	rw	**FFT Disable Request Bit** Used for enable/disable control of the FFT Engine.
DISS	1	rh	**FFT Disable Status Bit** Bit indicates the current status of the FFT Engine. DISABLED after reset.
0	31:2	r	**Reserved** Read as 0; should be written with 0.

29.1.9.2 FFT Module Identification Register

FFT Identification Register (FFT_ID)

The FFT identification register allows the programmer version-tracking of the module. The table below shows the identification register which is implemented in the FFT Engine.

FFT_ID
FFT Identification Register **(0008_H)** **Reset Value: $00E2\ C002_H$**



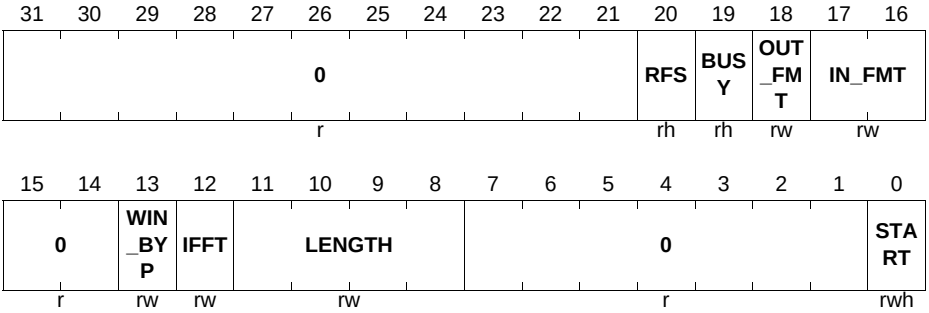
Field	Bits	Type	Description
MOD_REV	7:0	r	**Module Revision Number** This bit field defines the module revision number. 01_H : the value for max. 1K point engine 02_H : the value for max. 2K point engine.
MOD_TYPE	15:8	r	**Module Type** The bit field is set to $C0_H$ which defines the module as a 32-bit module.
MOD_NUMBER	31:16	r	**Module Number Value** This bit field defines a module identification number. The value for the FFT Engine is $00E2_H$.

29.1.9.3 FFT Control and Status Register

FFT Control and Status Register (FFT_CSR)

This is the main register to control FFT/IFFT operation and provide feedback on the current state of the FFT Engine.

FFT_CSR
FFT Control and Status Register (0040$_H$) **Reset Value: 0010 0000$_H$**



Field	Bits	Type	Description
START	0	rwh	**Start Transform** Start a new transform - cleared by hardware.
LENGTH	11:8	rw	**Length of Transform** Log base 2 of the required transform size. Length must be between 3 (transform of 8) and 11 (transform of 2048).
IFFT	12	rw	**Inverse FFT** 0_B perform FFT. 1_B perform IFFT.
WIN_ BYP	13	rw	**Window Bypass** 0_B use window function 1_B bypass window function
IN_FMT	17:16	rw	**Input Format** 0_D 1 complex 32-bit datum per 64-bit word 1_D 2 real 32-bit data per 64-bit word 2_D 2 complex 16-bit data per 64-bit word 3_D 4 real 16-bit data per 64-bit word
OUT_FMT	18	rw	**Output Format** 0_D 1 complex 32-bit datum per 64-bit word 1_D 2 complex 16-bit data per 64-bit word
BUSY	19	rh	**FFT Engine Busy** 0_D : FFT Engine Idle - no pending transforms 1_D : FFT Engine has at least one transform in operation.

Field	Bits	Type	Description
RFS	20	rh	**Ready For Start** 0_D : cannot accept a START operation 1_D : can accept a START operation.
0	31:21, 15:14, 7:1	r	**Reserved** Read as 0; should be written with 0

This register may be written at any time without setting the START bit, but this will have no effect on the FFT Engine. Only updates with the START bit set will change the FFT Engine state, and must be done before data can be loaded into the FFT Engine, to inform it of the input data formats and operational mode for that transform.

However the FFT Engine must be ready to accept a new START request. This register can be read and the RFS - ready for start - bit examined. If the RFS is true then a new transform can be started. A trigger will be sent to SRC_FFTERR if START is set when the FFT Engine is not ready. The commencement of new transforms in an interrupt driven manner is described in **Section 29.1.8**.

The IN_FMT field determines how data on the little-endian 64-bit data bus is loaded into the FFT Engine.

Table 29-4 Input Data Formats

IN_FMT	Halfword[3]	Halfword[2]	Halfword[1]	Halfword[0]
0	imag[0]		real[0]	
1	real[1]		real[0]	
2	imag[1]	real[1]	imag[0]	real[0]
3	real[3]	real[2]	real[1]	real[0]

The OUT_FMT field determines how data on the little-endian 64-bit data bus is unloaded from the FFT Engine.

Table 29-5 Output Data Formats

OUT_FMT	Halfword[3]	Halfword[2]	Halfword[1]	Halfword[0]
0	imag[0]		real[0]	
1	imag[1]	real[1]	imag[0]	real[0]

The BUSY bit reports the status of the FFT engine. If BUSY is set then at least one transform has either been requested with a START or is executing or is waiting for unloading

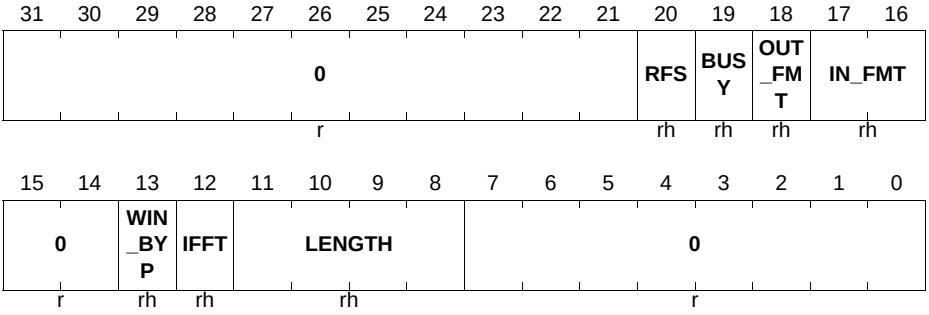
29.1.9.4 FFT History Registers

History registers allow the last few commands issued to the CSR to be retained for debug, the registers have no other effect upon the FFT Engine. They are reset to 0, and contain a non-zero value whenever they have been loaded by performing a write to the CSR register with START set. This write may or may not have setup a transform operation correclty depending upon the state of CSR.RFS at the time the START was issued.

FFT History Register 0 (FFT_HISTORY0)

The value of FFT CSR control fields when START was last issued.

FFT_HISTORY0
FFT History0 Register **(0060_H)** **Reset Value: $0000\ 0000_H$**



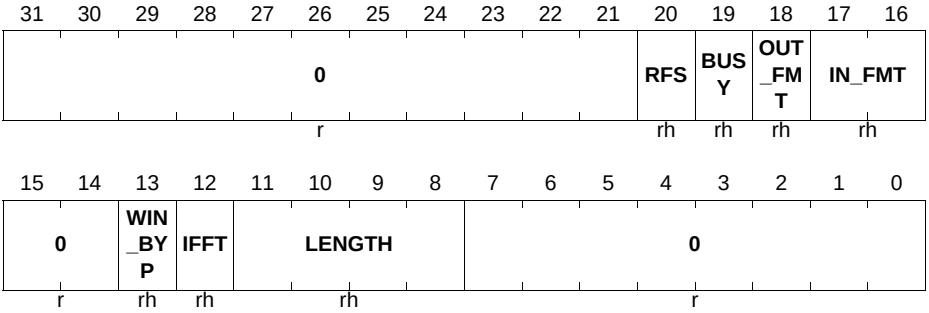
Field	Bits	Type	Description
LENGTH	11:8	rh	**Length of Transform** value of FFT_CSR.LENGTH when START command was last issued.
IFFT	12	rh	**Inverse FFT** value of FFT_CSR.IFFT when START command was last issued.
WIN_BYP	13	rh	**Window Bypass** value of FFT_CSR.WIN_BYP when START command was last issued.
IN_FMT	17:16	rh	**Input Format** value of FFT_CSR.IN_FMT when START command was last issued.

Field	Bits	Type	Description
OUT_FMT	18	rh	**Output Format** value of FFT_CSR.OUT_FMT when START command was last issued.
BUSY	19	rh	**FFT Engine Busy** value of FFT_CSR.BUSY when START command was last issued.
RFS	20	rh	**Ready For Start** value of FFT_CSR.RFS when START was last issued.
0	31:21, 15:14, 7:0	r	**Reserved** Always read as 0.

FFT History Register 1 (FFT_HISTORY1)

The value of FFT History Register 0 when START was last issued.

FFT_HISTORY1
FFT History1 Register **(0070_H)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
LENGTH	11:8	rh	**Length of Transform** value of FFT_HISTORY0.LENGTH when START command was last issued.
IFFT	12	rh	**Inverse FFT** value of FFT_HISTORY0.IFFT when START command was last issued.

Field	Bits	Type	Description
WIN_BYP	13	rh	**Window Bypass** value of FFT_HISTORY0.WIN_BYP when START command was last issued.
IN_FMT	17:16	rh	**Input Format** value of FFT_HISTORY0.IN_FMT when START command was last issued.
OUT_FMT	18	rh	**Output Format** value of FFT_HISTORY0.OUT_FMT when START command was last issued.
BUSY	19	rh	**FFT Engine Busy** value of FFT_HISTORY0.BUSY when START command was last issued.
RFS	20	rh	**Ready For Start** value of FFT_HISTORY0.RFS when START was last issued.
0	31:21, 15:14, 7:0	r	**Reserved** Always read as 0.

29.1.9.5 FFT Debug Control Registers

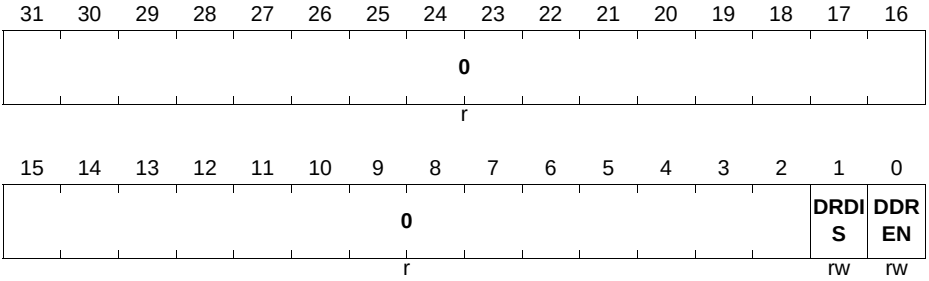
FFT OCDS Debug Access Register (FFT_ODA)

The FFT OCDS Debug Access (FFT_ODA) register is cleared by Debug Reset and by each System Reset when OCDS is disabled. It is not affected by System Reset when OCDS is enabled.

This register is used to control whether or not register bits which change state on read do so when read by the on-chip debug system.

Write access is 32 bit wide only and requires Supervisor Mode.

FFT_ODA
FFT OCDS Debug Access Register (00E4$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
DDREN	0	rw	**Destructive Debug Read Enable** *Note: For details see **Table 29-6*** 0_B Destructive read for debug read accesses are disabled 1_B Destructive read for debug read accesses are enabled
DRDIS	1	rw	**Destructive Read Disable** *Note: For details see **Table 29-6*** 0_B Destructive reads for read accesses are allowed. 1_B Destructive reads for read accesses are disabled
0	31:2	r	**Reserved** Read as 0; should be written with 0.

Table 29-6 Destructive Read Control encoding

DRDIS	DDREN	Description
0	0	Destructive read access are enabled for all masters except the OCDS master.
0	1	Destructive read access are enabled for all masters
1	-	Destructive read access are disabled for all masters

FFT OCDS Control and Status Register (FFT_OCS)

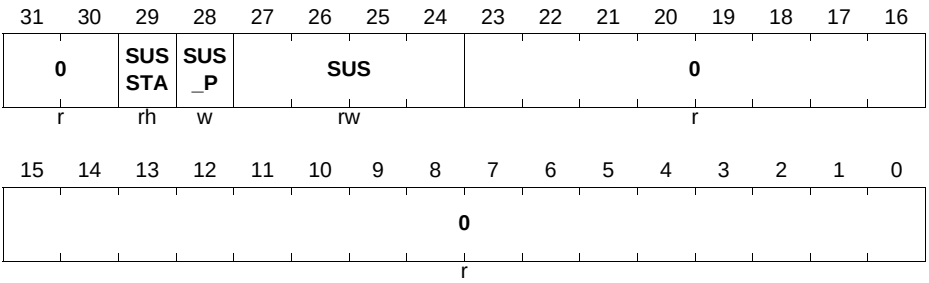
The FFT OCDS Control and Status (FFT_OCS) register is cleared by Debug Reset and by each System Reset when OCDS is disabled. It is not affected by System Reset when OCDS is enabled (signal eec_reset_n_i).

The FFT_OCS control register bits are only effective while the system is in debug mode. While not in debug mode, OCS reset values/modes are effective.

When a triggered by a high level on the suspend signal, the FFT/IFFT calculation will be halted as soon as practicable. Operation should resume when the suspend input is cleared.

Write access is 32 bit wide only and requires Supervisor Mode.

FFT_OCS
FFT OCDS Control and Status (00E8$_H$) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
SUS	27:24	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS). 0_H Will not suspend 1_H Reserved for Hard suspend which is not implemented. 2_H Soft suspend (Halt Mode) **others**, reserved
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended

Field	Bits	Type	Description
0	23:0, 31:30	r	**Reserved** Read as 0; must be written with 0.

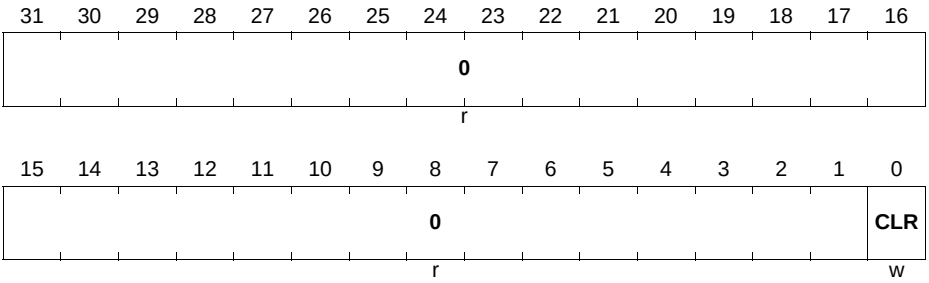
29.1.9.6 FFT Kernel Reset Registers

FFT Kernel Reset Status Clear Register (FFT_KRSTCLR)

The FFT Kernel Reset Status Clear register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

Write access is 32 bit wide only and requires Supervisor Mode.

FFT_KRSTCLR
FFT Kernel Reset Status Clear Register(00EC$_H$) **Reset Value: 0000 0000$_H$**



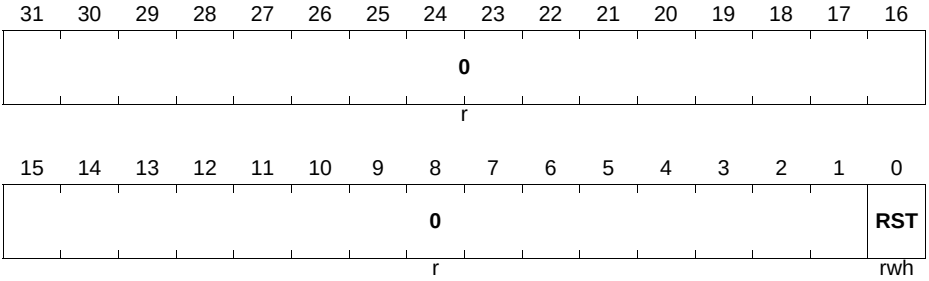
Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** *Note: Read always as 0_B.* 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT
0	31:1	r	**Reserved** Read as 0_B; should be written with 0_B.

FFT Kernel Reset Register 1 (FFT_KRST1)

The FFT Kernel Reset function is used to synchronously reset all the FFT Engine registers (except those connected to Debug reset) and the FFT Engine operation. To activate the kernel reset, it is necessary to set the RST bits by writing the value 1_B in both Kernel Reset Registers. The RST bit will be reset by hardware at the end of the kernel reset sequence.

Write access is 32 bit wide only and requires Supervisor Mode.

FFT_KRST1
FFT Kernel Reset Register 1 **($00F0_H$)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers are set. The RST bit will be cleared (re-set to 0_B) after the kernel reset is executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
0	31:1	r	**Reserved** Read as 0_B; should be written with 0_B.

FFT Kernel Reset Register 0 (FFT KRST0)

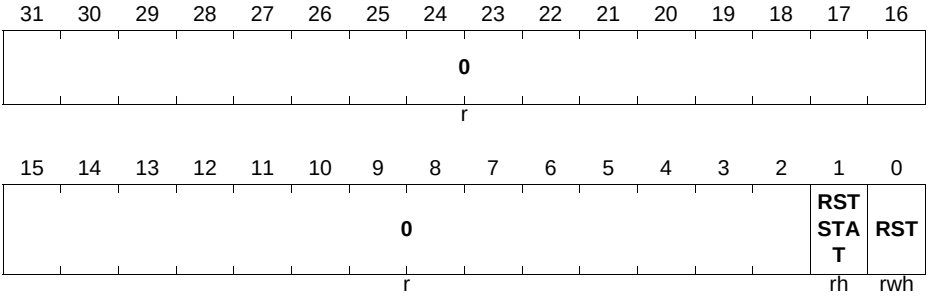
The FFT Kernel Reset function is used to synchronously reset all the FFT Engine registers (except those connected to Debug Reset) and the FFT Engine operation. To activate the kernel reset, it is necessary to set the RST bits by writing the value 1_B in both Kernel Reset Registers. The RST bit will be reset by hardware at the end of the kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to 1_B in the same clock cycle the RST bit is reset. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the KRSTCLR.CLR register bit.

During the execution of the kernel reset; until RSTSTAT is set, all write accesses to the kernel registers will result in an error acknowledge.

Write access is 32 bit wide only and requires Supervisor Mode.

FFT_KRST0
FFT Kernel Reset Register 0 (00F4$_H$) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. The RST bit will be cleared (reset to 0_B) b after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates whether a kernel reset was executed or not. This bit is set after the execution of a kernel reset in the same clock cycle both reset bits are cleared (FFT_KRST0.RST and FFT_KRST1.RST). This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved** Read as 0; should be written with 0.

29.2 Extended Memory (EMEM)

The Extended Memory (EMEM) contains RAM blocks (Tiles) which can be used as global memory. It has interfaces to the LMU and the BBB. The EMEM architecture is similar to the EMEM of the Emulation Devices (ED).

Applications:

- Data and program code storage
- FFT data buffering
- Image buffering

Features:

- 512 Kbyte RAM in total
- Concurrent access from LMU and BBB to different tiles possible
- BBB burst transfers supported
- ECC with SECDED (Single Error Correction, Double Error Detection)

29.2.1 Block Diagram

The Extended Memory consists of a regular memory array (ADM - Application Data Memory) build out of 64 Kbyte RAM Tiles. Additionally on certain products of the family the optional extension memories XAM is offered.

The Extended Memory provides two internal interfaces:

- LMU Interface
- BBB Interface with master and slave interface

The two interfaces allow a concurrent access to two different Tiles.

29.2.1.1 Extended Application Memory (XAM)

The XAM feature is a family concept for products, which require more than 1 MB Application RAM. The first EMEM RAM block (ADM) is partitioned into 16 Tiles. The following XAM EMEM block is without partitioning. A wrapper routes the accesses to ADM or XAM as a single continuous address range. The availability of XAM for different members of the product family is listed in **Table 29-7**.

29.2.1.2 Family Overview

Figure 29-4 shows an EMEM configuration example and **Table 29-7** the Extension Memory configurations for the different products of the family.

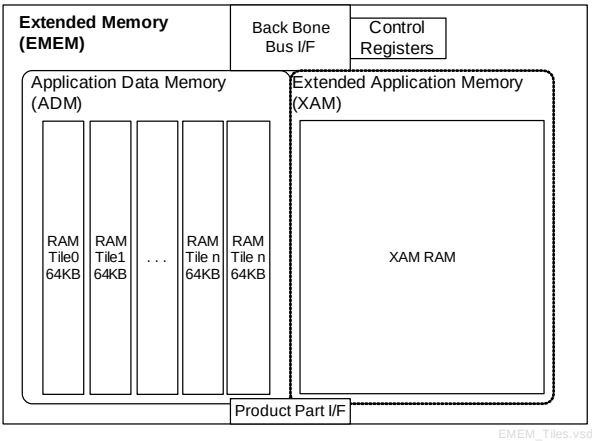


Figure 29-4 EMEM Configuration Example with ADM, XAM Tiles

Table 29-7 Extension Memory availability for the Product Family

Product	Total RAM Size	ADM Size	XAM Size	XAM Addr. LMU/BBB
TC29x	2048 KB	1 MB	1 MB	xx10 0000[1)]
TC27x	1024 KB	1 MB	-	-
TC26x	512 KB	512 KB	-	-
TC24x	256 KB	256 KB	-	-
TC23x	512 KB	512 KB	-	-

1) More information can be found in **Table 29-9**.

29.2.2 Operational Overview

The Extended Memory as a whole can be in Standby Locked Mode or in Enabled Mode.

Immediately after Power On Reset, the entire Extended Memory is in Unused Mode, thus the Tiles are not yet reserved for application mode. At any time, each Tile can be individually assigned to Application or again Unused Memory by setting its mode.

If a Tile has to be configured to Application Mode to activate the Extended Memory for the application.

This two step approach with selecting the global assignment for the Extended Memory of a Tile first and then operating it supports a safe operation.

Note: Before using the EMEM the boundary to this memory must be enabled by de-asserting the OSCU control bit OSTATE.EECDIS. Failing to do so will result in a hang-up of the CPU/DMA on the first read access to the EMEM.

Tile in Unused Mode

After Power On Reset, the Tiles are still in Unused Mode and are not yet reserved. They cannot be accessed before being assigned to Extended Memory by software. Later on, each Tile can be individually returned to Unused Memory space.

Tile in Application Mode

A RAM Tile in Application Mode can be accessed from the LMU or the BBB. The arbitration for concurrent accesses is done within the Extended Memory by delaying the BBB access.

Simultaneous Accesses

Up to two different Tiles can be accessed in parallel via the two interfaces, if these Tiles are configured in Application mode.

29.2.3 Extended Memory

By writing to Register **TILECONFIG**, unused Tiles can be assigned to Extended Memory. In Application Mode, write and read accesses can be performed by the LMU or BBB interfaces in arbitrary sequence. The arbitration is done internally with higher priority for the LMU interface.

29.2.4 Access Error Generation

If a Tile is accessed over an interface which is not supported in the current mode of the Tile, an error is signalled to the access initiator.

Any access attempt to RAM or a register during Standby Locked Mode is answered with an access error.

Table 29-8 Error Signaling for RAM Accesses

	LMU I/F	BBB I/F
Application Mode	Ok[1)]	Ok[1)]

1) If an ECC error is detected, an access error is signalled to the corresponding interface. If the width of a write access is smaller than the addressed memory word, an internal read-modify-write access is performed which potentially results in an ECC error for the read part. An ECC error is also generated, if a RAM location was not initialized before!

In Locked Mode, the registers **CLC**, **ID**, and **SBRCTR** are accessible via the LMU through the BBB master interface, access to other registers raises an error.

29.2.5 Tiles and RAM Address Ranges

The Extended Memory (ADM RAM area) can be accessed from two interfaces:

- LMU Interface
- BBB Interface

Accesses to the Extended Memory via the LMU are done by one of the CPUs or other SRI bus masters. To cater for cached/non-cached access the Extended Memory is mapped to two segments by the LMU.

The addressing of ADM and XAM depends on the interface. LMU and BBB have the same addressing view, just with a different base address.

The LMU/BBB addressing is based on the 1 MB ADM address area, followed by XAM always starting at the 1 MB address.

Assignment of Tiles to Trace Memory usually starts with Index 0 (**Table 29-9**).

Table 29-9 EMEM Tiles and Address Ranges

Tile	LMU[1)] TC24x	BBB	SBDAP Address
t; (ADM) t=0…7	$sF0t0000_H$ $sF0tFFFF_H$	$AF0t0000_H$ $AF0tFFFF_H$	$000t0000_H$ $000tFFFF_H$

1) $s = B_H$: non-cached access by CPU, $s = 9_H$: cached access by CPU

For register accesses see **Table 29-10**.

Figure 29-5 shows TC29x and TC26x as an example.

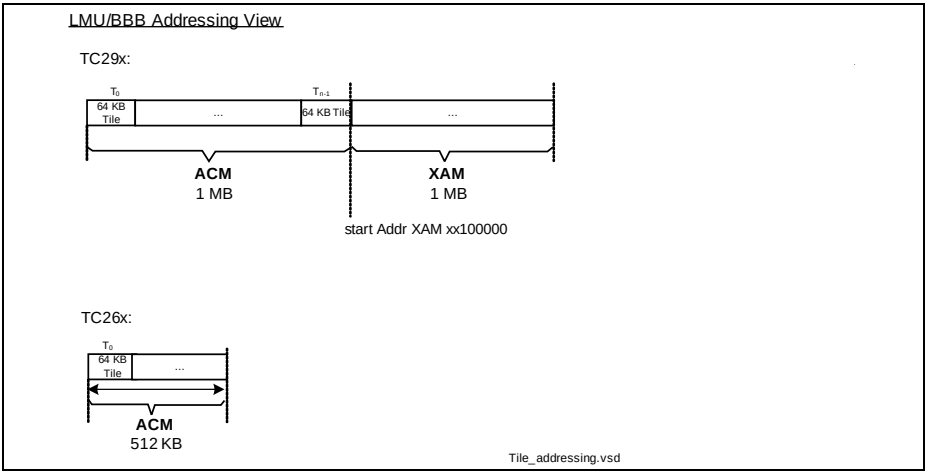


Figure 29-5 TCM and AXM interface addressing view.

More details about ADM and XAM addressing can be found in **Table 29-9**.

29.2.6 EMEM Initialization

After Power-on Reset the EMEM is locked and requires an unlock sequence similar to the EMEM of Emulation Device to get the memory initialized.

Unlocking the EMEM

To unlock EMEM, the following sequence is written to the **SBRCTR** register.

1. $0000\ 0002_H$,
2. $0000\ 0006_H$,
3. $0000\ 000E_H$

Writing any other value to **SBRCTR** before the complete sequence was written will reset the state machine, i.e. the sequence must be started again from the beginning. Writing to other registers however does not influence the sequence checker state machine.

The current state is signalled with bit **SBRCTR**.**STBLOCK**.

29.2.7 Usage Constraints

In this section assumptions are listed for using the EMEM module.

29.2.7.1 Clock Constraints

The clock for the BBB must not be faster than the EMEM clock. For the TC21x/TC22x/TC23x the LMU clock is equal to SRI clock.

29.2.7.2 BBB Interface

The BBB slave interface supports burst transfers, e.g. for image buffering.

29.2.8 Power Saving

In order to minimise power consumption, the clocks of inactive Tiles can be gated.

If clock gating is enabled (**SBRCTR**.ACGEN), all Tiles in Unused Mode are no longer clocked.

Tiles in Application are not gated directly. For this purpose they can first be set to Unused Mode.

The current clock gating status is individually indicated for each Tile (**SBRCTR**.ACGSTi).

29.2.9 EMEM Configuration

For power saving reasons, each tile is initially in unused mode. The register bits **TILECONFIG** allow to allocate the specific tiles as appliation memory.
The configuration state of each Tile is indicated in register **TILESTATE**.

29.2.10 EMEM Interfaces

As each Tile is arbitrated individually, transactions from the different interfaces into different Tiles are handled concurrently, i.e. up to two.

According to the word-lengths of the addressed RAM blocks, measures are taken for optimizing RAM accesses. Internal read-modify-write transactions are only required (due to ECC) if the word-length of data for the physical write access is smaller than the word-length of the memory.

29.2.10.1 LMU Interface

Read and write transactions are initiated by the LMU. Burst and read-modify-write transactions are not supported, but are split into basic and concatenated transactions.

Wide Read Transactions

Read accesses wider than the read-data word-length at the LMU are transformed into multi-beat transactions in subsequent cycles, taking into account wrapping around for unaligned addresses.

Concatenated read accesses of small words are not expected due to the wide-read access feature. If concatenated read accesses should occur anyhow, the EMEM processes them just like sequences of single read accesses without waiting for the master request to become inactive in between.

Concatenated Write Transactions

For write-widths smaller than the word-width of the memory, generally a complete memory word has to be synthesized before physically writing to memory, due to the required ECC which refers to complete memory words only. Whenever a write access is received from the LMU, a speculative read is immediately started and a ready signal returned to the LMU. In the next cycle, the read data is updated with the previous write data and written back, if the first write access is single. If instead another concatenated write transaction is received, the just obtained read data is not used, but the two subsequent write data portions are combined and written to memory. Thus, for a concatenated write, only the first of two consecutive write accesses causes a speculative internal read, because the protocol does not yet allow to decide for a first write, whether a second concatenated one is about to follow.

29.2.10.2 BBB Slave Interface

Transactions from the BBB to EMEM memory are initiated by any BBB master. In order to avoid internal read-modify-write accesses to EMEM-RAM, BBB write data received by the EMEM slave interface in burst operations is accumulated before being written into the memory. The EMEM control decides from the FPI opcode and the given memory word size whether a combination is possible or an internal read is required. These optimizations include unaligned burst transfers.

29.2.11 Feed-Through Transactions

The EEC must also be accessed from the product chip part. For this purpose, the EMEM feeds through transactions received from the LMU to the BBB Master Interface, if none of the EMEM's RAMs is addressed. Thus LMU and EMEM jointly act as bus-bridge between the SRI bus and the Backbone bus.

When acting as BBB Bus Master the EMEM tag and Supervisor Mode (SVM) is used.

29.2.11.1 EMEM Register Accesses

The SFRs of the EMEM are accessed via the BPI part of the BBB Slave Interface by any component connected to the BBB through a master interface, such as IOC32, and also the EMEM itself.

A register access from the product side is performed via LMU, the BBB Master Interface to the BBB, and from there via the BBB Slave Interface.

29.2.12 Address Map

All EMEM registers are prefixed "EMEM_" in the register map of the device.

EMEMoccupies an address window of 256 bytes. The relative register addresses in the register descriptions below are given as offset addresses to the base address of the window.

The base addresses for register accesses are identical for LMU and BBB. Internally in the EMEM, a window of 2 MB, addr[21:0] is available for addressing registers via BBB.

Note: This window is mapped to another segment when accessed via the SRI. Please, check production device address mapping to find the correct offset.

Access rights within the address range of EMEM:

- Read access to defined register addresses: U, SV
- Write access to defined register addresses: U, SV (no BE even if read-only register)
- Read access to empty addresses: No error (returns 0)
- Write access to empty addresses: BE
- Read or write when the kernel clock is turned off: BE (unless **CLC**)

Table 29-10 Registers Address Space

Module	Base Address	End Address	Note
EMEM	F90E 6000$_H$	F90E 60FF$_H$	CPU address

Table 29-11 Registers Overview

Register Short Name	Register Long Name	Offset Address	Page Number
CLC	Clock Control Register	00$_H$	Page 29-36
ID	Module Identification Register	08$_H$	Page 29-37
TILECONFIG	Tile Configuration Register	20$_H$	Page 29-33
TILESTATE	Tile Status Register	2C$_H$	Page 29-34
SBRCTR	Standby RAM Control Register	34$_H$	Page 29-34

29.2.13 EMEM Register Description

As detailed in the OCDS chapter, several reset domains exist on TC21x/TC22x/TC23x. All registers of the EMEM module belong to the Power On Reset domain and are reset by PORST. The **CLC** register additionally belongs to the EEC Reset domain.

TILECONFIG
Tile Configuration Register **(20_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		0		0		0		0		0		0		0	
r		r		r		r		r		r		r		r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T7		T6		T5		T4		T3		T2		T1		T0	
w		w		w		w		w		w		w		w	

Field	Bits	Type	Description
Tx **(x = 0 - 7)**	[2*x+1: 2*x]	w	**Tile x Allocation** 00_B Allocate tile 01_B Write has not effect. 10_B Reserved 11_B Unused *Note: The Tile allocation after reset requires “00” to be written to allocate the tile!*
0	[31:16]	r	**Reserved** returns 0 if read; should be written with 0.

TILESTATE
Tile Status Register **($2C_H$)** **Reset Value: 0000 FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		0		0		0		0		0		0		0	
r		r		r		r		r		r		r		r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TILE7		TILE6		TILE5		TILE4		TILE3		TILE2		TILE1		TILE0	
rh		rh		rh		rh		rh		rh		rh		rh	

Field	Bits	Type	Description
TILEx (x = 0 - 7)	[2*x+1: 2*x]	rh	**Usage of Tile x** 00_B Allocated 01_B Reserved 10_B Reserved 11_B Unused Mode
0	[31:16]	r	**Reserved** returns 0 if read; should be written with 0.

SBRCTR
Standby RAM Control Register **(34_H)** **Reset Value: 0000 0000$_H$**

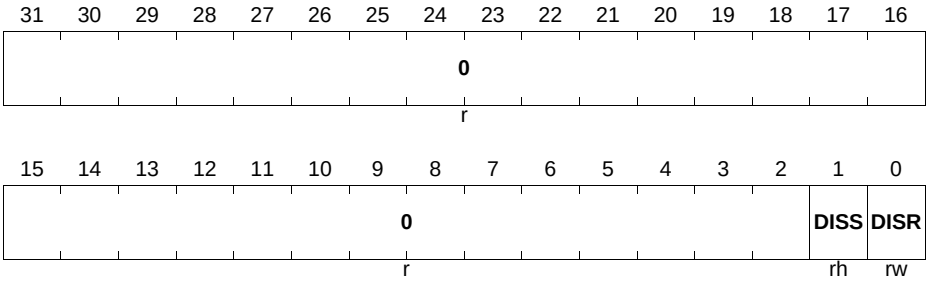
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	ACGST7	ACGST6	ACGST5	ACGST4	ACGST3	ACGST2	ACGST1	ACGST0
r	r	r	r	r	r	r	r	rh	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			ACGEN		0		0		STBSLK				STBULK		STBLOCK
r			rw		r		r		w				w		rh

Field	Bits	Type	Description
STBLOCK	0	rh	**Lock Flag** Shows the current lock state of the Extended Memory. 0_B Extended memory is locked 1_B Extended memory is unlocked
STBULK	[3:1]	w	**Unlock Lock Flag** In order to unlock the Extended Memory in Locked Mode in three consecutive[1] cycles the following pattern have to be written into this bit field: • 001_B • 011_B • 111_B At the same time 0 has to be written always into bit field STBSLK. If any of STBSLK is set when writing a non-zero pattern to STBULK, this is treated as invalid pattern and the Extended Memory in Locked Mode will not be unlocked. Reading this bit field always will return 0s
STBSLK	[7:4]	w	**Set Lock Flag** In order to lock the Extended Memory in operating mode 1001_B has to be written into this bit field. At the same time 0 has to be written into the bit field STBULK. If any of bits STBULK is set when writing 1001_B to STBSLK, this is treated as invalid pattern and the Lock Flag is not set. Reading this bit field always will return 0s
ACGEN	12	rw	**Automatic Clock Gating Enabling** 0_B Disabled 1_B Enabled
ACGSTx (x = 0 - 7)	x +16	rh	**Automatic Clock Gating Status of Tile x** 0_B Clock off 1_B Clock on *Note: These bits are cleared automatically when entering Locked Mode.*
0	[11:9], 8, 15, [14:13], [31:24]	r	**Reserved** returns 0 if read; should be written with 0.

1) Intermediate read or write cycles not accessing **SBRCTR** are allowed.

CLC
Clock Control Register **(00_H)** **Reset Value: 0000 0003$_H$**

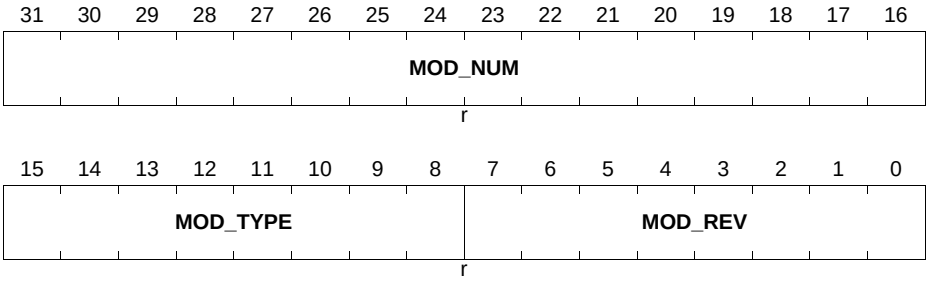


Field	Bits	Type	Description
DISR	0	rw	**Disable Request Bit** 0_B Module disable is not requested 1_B Module disable is requested
DISS	1	rh	**Disable Status Bit** 0_B Module enabled 1_B Module disabled
0	[31:2]	r	**Reserved** returns 0 if read; should be written with 0

The clock control register allows to switch the memory subsystem on or off. After Power On Reset the memory subsystem is disabled. The memory subsystem can be enabled by clearing bit DISR. It is ensured that the **CLC** is still writable when the clock is switched off.

ID

Module Identification Register (08$_H$) Reset Value: 00E0 C004$_H$



Field	Bits	Type	Description
MOD_REV	[7:0]	r	**Module Revision Number**
MOD_TYPE	[15:8]	r	**Module Type** C0$_H$ for 32 bit peripheral
MOD_NUM	[31:16]	r	**Module Number** For module identification

30 Ethernet MAC (ETH)

This chapter describes the Ethernet MAC [1) 2)] of the TC21x/TC22x/TC23x.

30.1 Overview

Figure 30-1 shows the basic structure of the module:

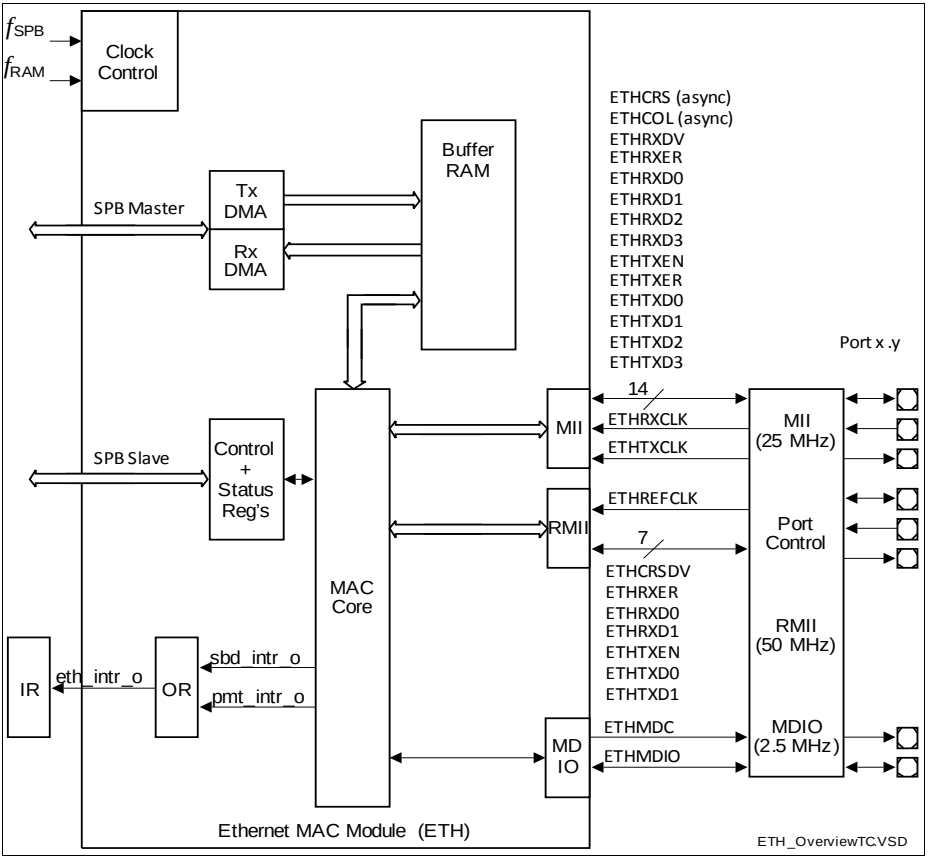


Figure 30-1 Ethernet MAC Module Overview

1) Excerpted portions are Synopsys Proprietary. Used with permission.

2) Module is not available in some variants. In these variants registers are still accessible but functionality cannot be guaranteed.

30.1.1 General Module Description

The DWC Ether MAC 10/100/1000 Universal, V3.7a, commonly referred to as GMAC-UNIV in this document, enables a host to transmit and receive data over Ethernet in compliance with the IEEE 802.3-2002 standard. In TC21x/TC22x/TC23x it is configured to support 10 and 100 Mbit modes. Note that Gigabit mode is not configured, nevertheless the core is referred to as GMAC to avoid confusion with other cores.

The GMAC-UNIV provides an optimized (with respect to gate count and latency), configurable, flexible product to meet the needs of various applications and customers, and supports a multitude of industry standard interfaces to the PHY: Media Independent Interface (MII) defined in the IEEE 802.3 specifications and Reduced Media Independent Interface (RMII).The GMAC-AHB is designed to interface to the industry standard AMBA High-Performance Bus (AHB) on the application side. In TC21x/TC22x/TC23x an adaptation logic interfaces to the SPB bus.

The GMAC-UNIV is compliant to the following standards:

- IEEE 802.3-2002 for Ethernet MAC
- IEEE 1588-2008 standard for precision networked clock synchronization
- RMII specification from RMII consortium
- MII specification IEEE 802.3

Note: Although the naming convention used in this databook and the source code indicate “GMAC,” the databook is also applicable to the 10/100 Universal product.

30.1.2 System Overview

30.1.2.1 System-Level Block Diagram

A system-level block diagram is shown in **Figure 30-2**.

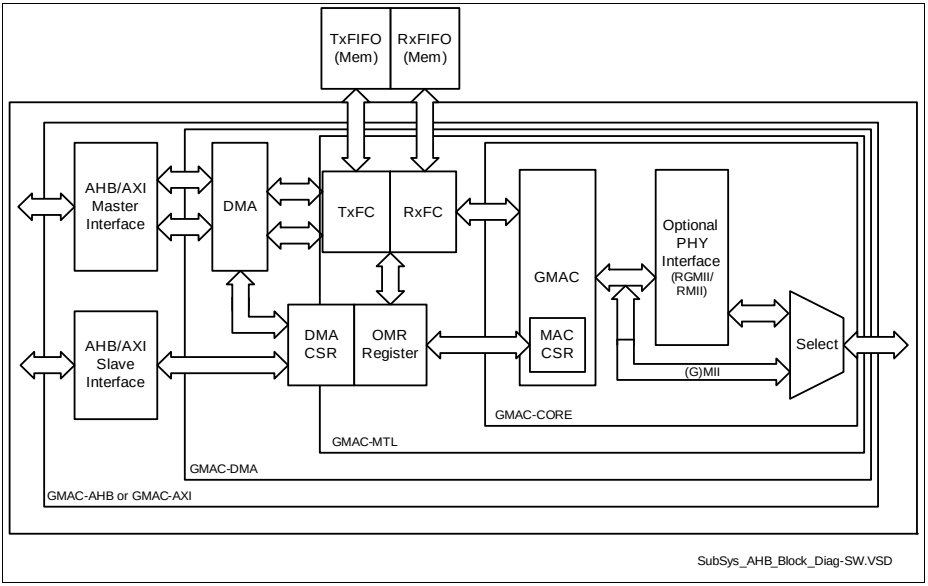


Figure 30-2 GMAC-UNIV Block Diagram

30.1.2.2 Interfaces

The GMAC-AHB transfers data to system memory through the AHB master interface. The host CPU uses the default 32-bit AHB Slave interface to access the GMAC subsystem's Control and Status registers (CSRs).

The GMAC-UNIV supports the following PHY interfaces:

- Media Independent Interface (MII)
- Reduced MII (RMII)

30.1.2.3 Transmit and Receive FIFOs

The Transmit FIFO (Tx FIFO) buffers data read from system memory by the DMA before transmission by the GMAC Core. Similarly, the Receive FIFO (Rx FIFO) stores the Ethernet frames received from the line until they are transferred to system memory by

the DMA. These are asynchronous FIFOs, as they also transfer the data between the application clock and the GMAC line clocks.

Tx/Rx FIFOs are 32 bits wide and 4/4 kByte deep.

30.1.3 Features List

The GMAC-UNIV includes the following features, listed by category.

30.1.3.1 GMAC Core Features

- Supports 10/100-Mbit/s data transfer rates with the following PHY interfaces
  - IEEE 802.3-compliantRMII/MII (default) interface to communicate with an external Fast Ethernet PHY
- Supports both full-duplex and half-duplex operation
  - Supports CSMA/CD Protocol for half-duplex operation
  - Supports IEEE 802.3x flow control for full-duplex operation
  - Optional forwarding of received pause control frames to the user application in full-duplex operation
  - Back-pressure support for half-duplex operation
  - Automatic transmission of zero-quanta pause frame on deassertion of flow control input in full-duplex operation
- Preamble and start-of-frame data (SFD) insertion in Transmit, and deletion in Receive paths
- Automatic CRC and pad generation controllable on a per-frame basis
- Options for Automatic Pad/CRC Stripping on receive frames
- Programmable frame length to support Standard or Jumbo Ethernet frames with sizes up to 16 KB
- Programmable InterFrameGap (40-96 bit times in steps of 8)
- Supports a variety of flexible address filtering modes:
  - Up to 31 additional 48-bit perfect (DA) address filters with masks for each byte
  - Up to 31 48-bit SA address comparison check with masks for each byte
  - 64-bit Hash filter (optional) for multicast and uni-cast (DA) addresses
  - Option to pass all multicast addressed frames
  - Promiscuous mode support to pass all frames without any filtering for network monitoring
  - Passes all incoming packets (as per filter) with a status report
- Separate 32-bit status returned for transmission and reception packets
- Supports IEEE 802.1Q VLAN tag detection for reception frames
- Separate transmission, reception, and control interfaces to the Application
- Supports 32-bit data transfer interface on the system-side
- Complete network statistics (optional) with RMON/MIB Counters (RFC1757/RFC2819/RFC2665). It is completely under control of higher protocol level (SW) to make use of these counters.
- MDIO Master interface for PHY device configuration and management, e.g. for switching the PHY in external loopback mode.

- Optional module for detection of LAN wake-up frames and AMD Magic Packet frames
- Optional Receive module for checksum off-load for received IPv4 and TCP packets encapsulated by the Ethernet frame
- Optional Enhanced Receive module for checking IPv4 header checksum and TCP, UDP, or ICMP checksum encapsulated in IPv4 or IPv6 datagrams.
- Optional module to support Ethernet frame time stamping as described in IEEE 1588-2008. Sixty-four-bit time stamps are given in each frame's transmit or receive status.

30.1.3.2 DMA Block Features

The DMA block exchanges data between the MTL block and host memory. A set of registers (DMA CSR) to control DMA operation is accessible by the host.

DMA features include:

- 32-bit data transfers
- Single-channel Transmit and Receive engines
- Fully synchronous design operating on a single system clock (except for CSR module, when a separate CSR clock is configured)
- Optimization for packet-oriented DMA transfers with frame delimiters
- Byte-aligned addressing for data buffer support
- Dual-buffer (ring) or linked-list (chained) descriptor chaining
- Descriptor architecture, allowing large blocks of data transfer with minimum CPU intervention; each descriptor can transfer up to 8 KB of data
- Comprehensive status reporting for normal operation and transfers with errors
- Individual programmable burst size (SINGLE, INCR4/8, not supported by the System: INCR of undefined length) for Transmit and Receive DMA Engines for optimal host bus utilization
- Programmable interrupt options for different operational conditions
- Per-frame Transmit/Receive complete interrupt control
- Round-robin or fixed-priority arbitration between Receive and Transmit engines
- Start/Stop modes
- Separate ports for host CSR access and host data interface

30.1.3.3 Transaction Layer (MTL) Features

The MTL block consists of two sets of FIFOs: a Transmit FIFO with programmable threshold capability, and a Receive FIFO with a configurable threshold (default of 64 bytes).

MTL features include:

- 32--bit Transaction Layer block providing a bridge between the application and the GMAC-CORE

- Single-channel Transmit and Receive engines
- Data transfers executed using simple FIFO-protocol
- Synchronization for all clocks in the design (Transmit, Receive and system clocks)
- Optimization for packet-oriented transfers with frame delimiters
- Four Separate ports for system-side and GMAC-CORE-side transmission and reception
- Two RAM-based asynchronous FIFOs with synchronous/asynchronous Read and Write operation with respect to the Read and Write clocks (one for transmission and one for reception)
- Receive Status vectors inserted into the Receive FIFO after the EOF transfer enables multiple-frame storage in the Receive FIFO without requiring another FIFO to store those frames' Receive Status.
- Configurable Receive FIFO threshold (default fixed at 64 bytes) in Cut-Through mode
- Option to filter all error frames on reception and not forward them to the application in Store-and-Forward mode
- Option to forward under-sized good frames
- Supports statistics by generating pulses for frames dropped or corrupted (due to overflow) in the Receive FIFO
- Supports Store and Forward mechanism for transmission to the GMAC core
- Supports threshold control for transmit buffer management
- Supports configurable number of frames to be stored in FIFO at any time. The default is up to 8 frames in GMAC-MTL configuration.
- Automatic generation of PAUSE frame control or backpressure signal to the GMAC core based on Receive FIFO-fill (threshold configurable) level.
- Handles automatic retransmission of Collision frames for transmission
- Discards frames on late collision, excessive collisions, excessive deferral and underrun conditions
- Software control to flush Tx FIFO
- Data FIFO RAM chip-select disabled when inactive, to reduce power consumption
- Optional module to calculate and insert IPv4 header checksum and TCP, UDP, or ICMP checksum in frames transmitted in Store-and-Forward mode.

30.1.3.4 Monitoring, Test, and Debugging Support Features

- Supports internal loopback on the MII for debugging
- external loopback is supported via the integrated MDIO controlling the PHY
- DMA states (Tx and Rx) given as status bits
- Debug status register that gives status of FSMs in Transmit and Receive data-paths and FIFO fill-levels.
- Application Abort status bits
- MMC (RMON) module in the GMAC core
- Current Tx/Rx Buffer pointer as status registers
- Current Tx/Rx Descriptor pointer as status registers

30.2 Architecture

30.2.1 Introduction

The GMAC-UNIV is a highly configurable product with multiple interfaces to the system side or the PHY side. Major modules and functions are only selected when required. A block diagram of the GMAC-UNIV's system configuration is provided in **Figure 30-3**.

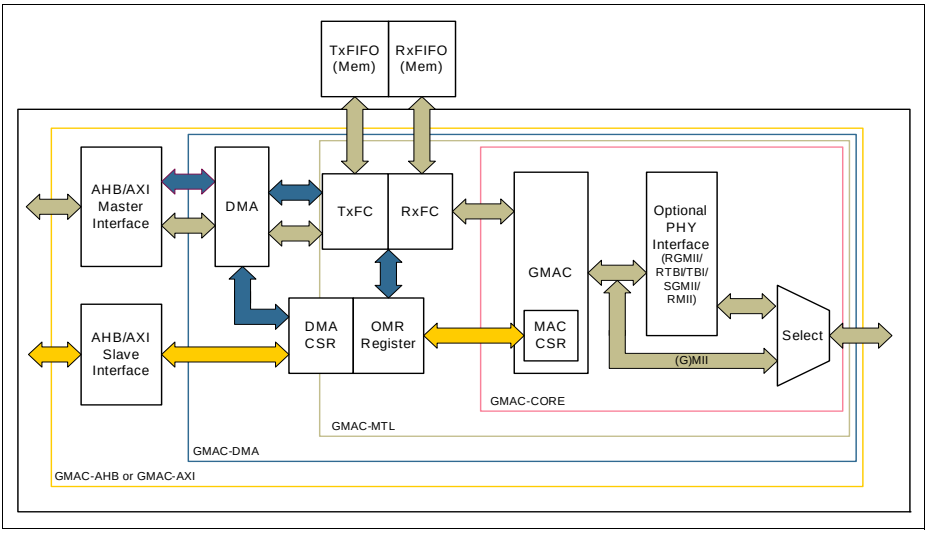


Figure 30-3 GMAC-UNIV Block Diagram

The following sections describe the functionality of the major blocks and timing behavior of the interfaces.

- **Section 30.2.2** provides an overview of precision network time synchronization using the Precision Time Protocol established under the IEEE 1588 standard.
- **Section 30.2.3** describes the functionality of the AHB interfaces in GMAC-AHB configuration
- **Section 30.2.4** describes the functioning of the DMA.
- **Section 30.2.5** explains the functioning of the MAC Transaction Layer (MTL) module which controls the Receive and Transmit FIFO memories.
- **Section 30.2.6** describes the functionality of the (G)MAC core transmitter and receiver.
- **Section 30.2.7**, **Section 30.2.8**, and **Section 30.2.8** describe the optional RMON, Power Management, and SMA interface modules.
- **Section 30.2.11** describes the Interrupt signal structure in the (G)MAC core.

30.2.2 IEEE 1588-2002 Overview

The IEEE 1588 standard defines a protocol enabling precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects. The protocol applies to systems communicating by local area networks supporting multicast messaging, including (but not limited to) Ethernet. This protocol enables heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. The protocol supports system-wide synchronization accuracy in the sub-microsecond range with minimal network and local clock computing resources.

The messaged-based protocol, named Precision Time Protocol (PTP), is transported over UDP/IP. The system or network is classified into Master and Slave nodes for distributing the timing/clock information. The protocol's technique for synchronizing a slave node to a master node by exchanging PTP messages is depicted in **Figure 30-4**.

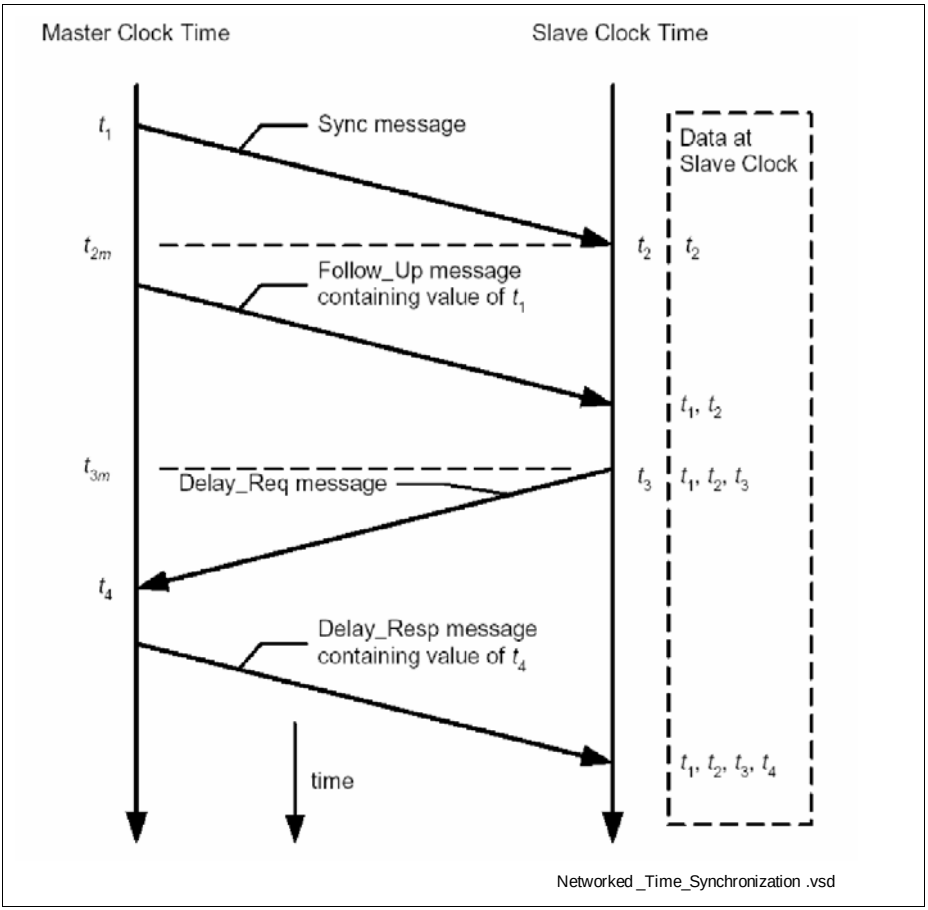


Figure 30-4 Networked Time Synchronization

1. The master broadcasts PTP Sync messages to all its nodes. The Sync message contains the master's reference time information. The time at which this message leaves the master's system is t1 and must, for Ethernet ports, be captured at the MII.
2. A slave receives the Sync message and also captures the exact time, t2, using its timing reference.
3. The master then sends the slave a Follow_up message, which contains t1 information for later use.
4. The slave sends the master a Delay_Req message, noting the exact time, t3, at which this frame leaves the MII.

5. The master receives this message, capturing the exact time, t4, at which it enters its system.
6. The master sends the t4 information to the slave in the Delay_Resp message.
7. The slave uses the four values of t1, t2, t3, and t4 to synchronize its local timing reference to the master's timing reference.

Most of the protocol implementation occurs in the software, above the UDP layer. As described above, however, hardware support is required to capture the exact time when specific PTP packets enter or leave the Ethernet port at the MII. This timing information must be captured and returned to the software for the proper implementation of PTP with high accuracy.

30.2.2.1 Reference Timing Source

To get a snapshot of the time, the core requires a reference time in 64-bit format (split into two 32-bit channels, with the upper 32-bits providing time in seconds, and the lower 32-bits indicating time in nanoseconds) as defined in the IEEE 1588 specification.

Internal Reference Time

This takes only the reference clock input and uses it to generate the Reference time (also called the System Time) internally and use it to capture time stamps. The generation, update, and modification of the System Time are described in **System Time Register Module**.

30.2.2.2 Transmit Path Functions

When a frame's SFD is output on the MII, a time stamp is captured. Frames for which capturing a time stamp is required are controllable on a per-frame basis. In other words, each transmit frame can be marked to indicate whether or not a time stamp must be captured for that frame.

No snooping or processing of the transmitted frames is performed to identify PTP frames. Framewise control is exercised through control bits in the transmit descriptor (as described in **Descriptor Format With IEEE 1588 Time Stamping Enabled**).

Captured time stamps are returned to the application in a manner similar to that in which status is provided for frames. The time stamp is returned to software inside the corresponding transmit descriptor, thus connecting the time stamp automatically to the specific PTP frame. The 64-bit time stamp information is written back to the TDES2 and TDES3 fields, with TDES2 holding the time stamp's 32 least significant bits, except as described in **Transmit Time Stamp Field**.

Note: When the alternate (enhanced) descriptor is selected, the 64-bit time-stamp is written in TDES6 and TDES7, respectively

30.2.2.3 Receive Path Functions

When the IEEE 1588 time-stamping feature is selected and enabled, the Ethernet MAC captures the time stamp of all frames received on the MII. No snooping or processing of the received frames is performed to identify PTP frames in the default mode (Advanced Time Stamp feature is not selected).

The core returns the time-stamp to the software in the corresponding receive descriptor. The 64-bit time stamp information is written back to the RDES2 and RDES3 fields, with RDES2 holding the time stamp's 32 least significant bits, except as mentioned in **Receive Time Stamp** The time stamp is only written to the receive descriptor for which the Last Descriptor status field has been set to 1 (the EOF marker). When the time stamp is not available (for example, due to an RxFIFO overflow) an all-ones pattern is written to the descriptors (RDES2 and RDES3), indicating that time stamp is not correct. If the software uses a control register bit to disable time stamping, the DMA does not alter RDES2 or RDES3.

Note: When the alternate (enhanced) descriptor is selected, the 64-bit time-stamp is written in RDES6 and RDES7, respectively. RDES0[7] will indicate whether the time-stamp is updated in RDES6/7 or not.

30.2.2.4 Time Stamp Error Margin

According to the IEEE 1588 specifications, the time stamp must be captured at the SFD of transmitted and received frames at the MII interface. Since the reference timing source (the PTP clock, clk_ptp_ref_i) is different from the MII clocks, a small error margin is introduced, due to the transfer of information across asynchronous clock domains.

In the transmit path, the captured and reported time stamp has a maximum error margin of 2 PTP clocks. In other words, the captured time stamp has the value of the reference time source given within 2 clocks after the SFD has been transmitted on the MII.

Similarly, on the receive path, the error margin is 3 MII clocks, plus up to 2 PTP clocks. You can ignore the error margin due to the 3 MII clocks by assuming that this constant delay is present in the system (or link) before the SFD data reaches the GMAC's MII interface.

30.2.2.5 Frequency Range of Reference Timing Clock

Because asynchronous logic is in place for time stamp information transfers across clock domain, a minimum delay is required between two consecutive time stamp captures. This delay is 3 clock cycles of both the MII and PTP clocks. If the gap is shorter, the GMAC does not take a time stamp snapshot for the second frame.

The maximum PTP clock frequency is limited by the maximum resolution of the reference time and the timing constraints achievable for logic operating on the PTP clock. Another factor to consider is that the resolution, or granularity, of the reference time source determines the accuracy of the synchronization. Hence, a higher PTP clock

frequency gives better system performance. The minimum PTP clock frequency depends on the time required between two consecutive SFD bytes. Because the MII clock frequency is fixed by IEEE specification, the minimum PTP clock frequency required for proper operation depends on the core's operating mode and operating speed.

For example, in 100 Mbit/s full-duplex operation, the minimum gap between two SFDs is 160 MII clocks (128 clocks for a 64-byte frame + 24 clocks of min IFG + 8 clocks of preamble).

In the example, (3 x PTP) + (3 x MII) ≤ 160 x MII; thus, the minimum PTP clock frequency is about 0.5 MHz ((160 - 3) x 40 ns / 3 = 2.093 ns period)

30.2.2.6 Advanced Time Stamp Feature Support

In addition to the basic features for time stamp mentioned in **IEEE 1588-2002 Overview**, the advanced time stamp option has the following features.

- Support for the IEEE 1588-2008 (Version 2) timestamp format.
- Option to take snapshot for all frames or for PTP type frames.
- Option for taking snapshot for event messages only.
- Option to take the snapshot based on the clock type (ordinary, boundary, end-to-end and peer-to-peer)
- Option to select the node to be a Master or Slave for ordinary and boundary clock.
- Identification of PTP message type, version, and PTP payload sent directly over Ethernet given as status.
- Option to measure time in digital or binary format.

Peer-to-Peer PTP (Pdelay) Transparent Clock (P2P TC) Message Support

The IEEE 1588-2008 version supports Pdelay message in addition to SYNC, Delay Request, Follow-up and Delay Response messages. **Figure 30-5** shows the method to calculate the propagation delay in clocks supporting peer-to-peer path correction.

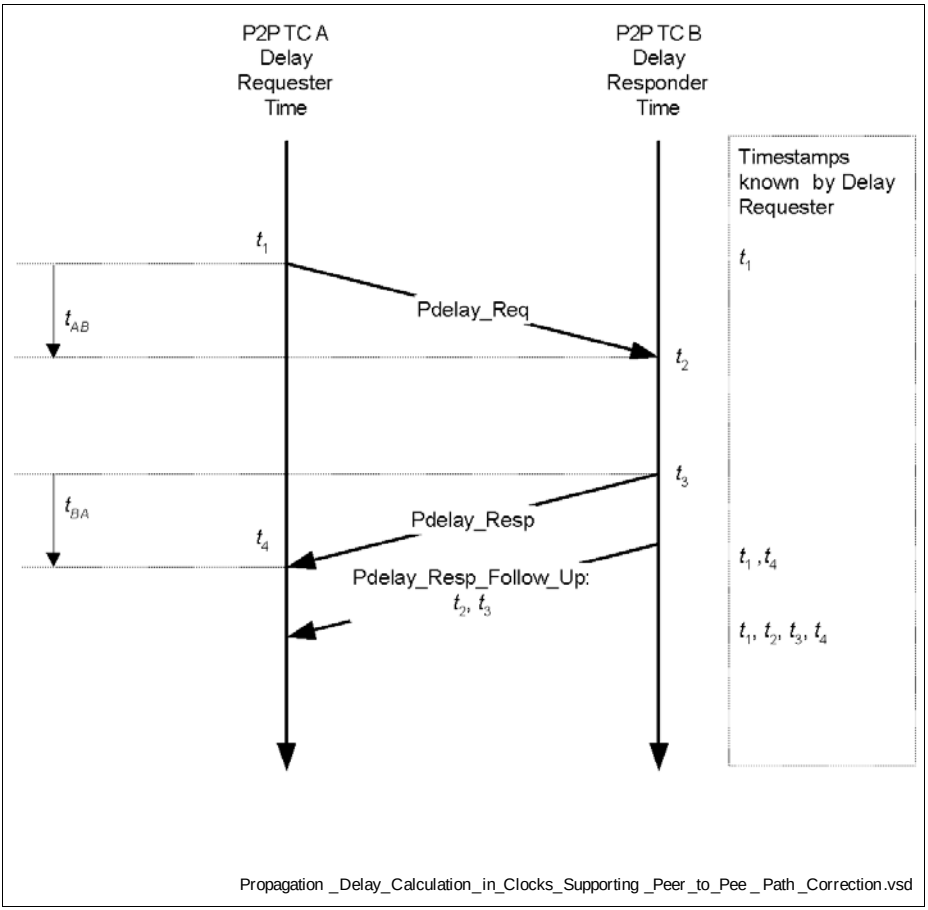


Figure 30-5 Propagation Delay Calculation in Clocks Supporting Peer-to-Peer Path Correction

The link delay measurement starts with port-1 issuing a “Pdelay_Req” message and generating a timestamp, for the Pdelay_Req message. Port-2 receives the “Pdelay_Req” message and generates a timestamp, t2, for this message. Port-2 returns a Pdelay_Resp message and generates a timestamp, t3, for this message. To minimize errors due to any frequency offset between the two ports, Port-2 returns the Pdelay_Resp message as quickly as possible after the receipt of the Pdelay_Req message.

Port-2 either:

- Returns the difference between the timestamps t2 and t3 in the Pdelay_Resp message,
- Returns the difference between the timestamps t2 and t3 in a Pdelay_Resp_Follow_Up message, or
- Returns the timestamps t2 and t3 in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages respectively.

Port-1 generates a timestamp, t4, upon receiving the Pdelay_Resp message. Port-1 then uses these four timestamps to compute the mean link delay.

Clock Types

The type of clock nodes supported in IEEE 1588-2008 is described in this section. The corresponding support provided by the advanced time stamp feature for each of the clock type is also mentioned.

1. Ordinary clock support: In this type the clock can be a grandmaster or a slave clock. This clock has a single PTP state.
   Table 30-1 shows the messages for which time-stamp snapshot is taken on the receive side for Master and Slave nodes.
   The ordinary clock in the domain supports a single copy of the protocol and has a single PTP state and will typically be a single physical port. In typical industrial automation applications, an ordinary clock is associated with an application device such as sensors and actuators. In telecom applications, the ordinary clock may be associated with a timing demarcation device.
   Typically for ordinary clock, you will need to take snapshot for only one type of PTP messages. For e.g. you will require supporting either version 1 or 2 PTP messages, not both.
   The following features are supported.
   a) Sends and receives PTP messages. The time stamp snapshot can be controlled as described in **32-bit Register - Timestamp_Control**.
   b) Maintains the data sets (e.g., time stamp values).
2. Boundary clock support: This type of clock is similar to the ordinary clock except for the following.
   Hence the features of ordinary clock holds good for the boundary clock also.
   The boundary clock typically has several physical ports communicating with the network. The messages related to synchronization, master-slave hierarchy and signaling terminate in the protocol engine of the boundary clock and are not forwarded. The PTP message type status given by the core (refer to **Receive Path Functions**) will help you to quickly identify the type of message and take appropriate action.
   a) The clock data sets are common to all ports of the boundary clock
   b) The local clock is common to all ports of the boundary clock.
3. End to end transparent clock support: The end-to-end transparent clock forwards all messages like normal bridge, router or repeater. The resident time needs to be

computed to update the correctionField. Hence snapshot needs to be taken for the messages mentioned in **Table 30-2**.

In the end-to-end transparent clock, the residence times are accumulated in a special field (correctionField) of the PTP event (SYNC) message or the associated Follow-up (FOLLOW_UP) Message. Hence it is important to take a snapshot for these messages alone. This can be quickly done by setting the control bit (TSEVNTENA), which enables snapshot to be taken for event messages and also selecting the type of clock in the “Time Stamp Control Register”.

The residence time is also corrected for Delay_Req messages (but snapshot of the timestamp is not required). The message type statuses provided helps you to quickly identity the message and update the correctionField.

The message type status provided will also help in taking appropriate action depending on the type of PTP message received.

4. Peer to peer transparent clock support: In this type of clock the computation of the link delay is based on an exchange of Pdelay_Req, Pdelay_Resp and Pdelay_Resp_Follow_Up messages with the link peer. Hence support for taking snapshot for the event messages related to Pdelay is added. **Table 30-3**.

   The transparent clock corrects only the SYNC and Follow-up message. As discussed earlier this can be achieved using the message status provided.

   The type of clock to be implemented will be configurable through control register, as mentioned in **32-bit Register - Timestamp_Control**. To ensure that the snapshot is taken only for the messages indicated in the table for the corresponding clock type, the “TSEVNTENA: Enable Time Stamp Snapshot for Event Messages” bit has to be set.

Table 30-1 PTP Messages for which Snapshot is Taken on Receive Side for Ordinary Clock

Master	Slave
Delay_Req	SYNC

Table 30-2 PTP Messages for which Snapshot is Taken for Transparent Clock Implementation

SYNC
FOLLOW_UP

Table 30-3 PTP Messages for which Snapshot is Taken for Peer-to-Peer Transparent Clock Implementation

SYNC
Pdelay_Req
Pdelay_Resp

PTP Processing and Control

The common message header for PTP messages is shown below. This format is taken from IEEE standard 1588-2008 (Revision of IEEE Std. 1588-2002).

Table 30-4 Message Format Defined in IEEE 1588-2008

BITS		OCTETS	OFFSET
transportSpecific	messageType	1	0
Reserved	versionPTP	1	1
messageLength		2	2
domainNumber		1	4
Reserved		1	5
flagField		2	6
correctionField		8	8
Reserved		4	16
sourcePortIdentity		10	20
sequenceId		2	30
controlField[1)]		1	32
logMessageInterva		1	33

1) controlField is used in version 1. For version 2, messageType field will be used for detecting different message types.

There are some fields in the PTP frame that are used to detect the type and control the snapshot to be taken. This is different for PTP frames sent directly over Ethernet, PTP frames sent over UDP / IPv4 and PTP frames that are sent over UDP / IPv6. The following sections provide information on the fields that are used to control taking the snapshot.

PTP Frame Over IPv4

Table 30-5 gives the details of the fields that will be matched to control snapshot for PTP packets over UDP over IPv4 for IEEE 1588 version 1 and 2. Note that the octet positions

for tagged frames will be offset by 4. This is based on Appendix-D of the IEEE 1588-2008 standard and the message format defined in **Table 30-4**.

Table 30-5 IPv4-UDP PTP Frame Fields Required for Control and Status

Field Matched	Octet Position	Matched Value	Description
MAC Frame type	12, 13	0x0800	IPv4 datagram
IP Version and Header Length	14	0x45	IP version is IPv4
Layer-4 protocol	23	0x11	UDP
IP Multicast address (IEEE 1588 version 1)	30, 31, 32, 33	0xE0,0x00, 0x01,0x81 (or 0x82 or 0x83 or 0x84)	Multicast IPv4 addresses allowed. 224.0.1.129 224.0.1.130 224.0.1.131 224.0.1.132
IP Multicast address (IEEE 1588 version 2)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (Hex) 0xE0, 0x00, 0x00, 0x6B (Hex)	PTP-primary multicast address: 224.0.1.129 PTP-Pdelay multicast address: 224.0.0.107
UDP destination port	36, 37	0x013F, 0x0140	0x013F – PTP event message[1)] 0x0140 – PTP general messages
PTP control field (IEEE version 1)	74	0x00/0x01/0x02 /0x03/0x04	0x00 – SYNC, 0x01 – Delay_Req, 0x02 – Follow_Up 0x03 – Delay_Resp 0x04 – Management

Table 30-5 IPv4-UDP PTP Frame Fields Required for Control and Status (cont'd)

Field Matched	Octet Position	Matched Value	Description
PTP Message Type Field (IEEE version 2)	42 (nibble)	0x0/0x1/0x2/0x3/0x8/0x9/0xB/0xC/0xD	0x0 – SYNC 0x1 – Delay_Req 0x2 – Pdelay_Req 0x3 – Pdelay_Resp 0x8 – Follow_Up 0x9 – Delay_Resp 0xA – Pdelay_Resp_Follow_Up 0xB – Announce 0xC – Signaling 0xD - Management
PTP version field	43 (nibble)	0x1 or 0x2	0x1 – Supports PTP version 1 0x2 – Supports PTP version 2

1) PTP event messages are SYNC, Delay_Req (IEEE 1588 version 1 and 2) or Pdelay_Req, Pdelay_Resp (IEEE 1588 version 2 only).

PTP Frame Over IPv6

Table 30-6 gives the details of the fields that will be matched to control snapshot for PTP packets over UDP over IPv6 for IEEE 1588 version 1 and 2. Note that the octet positions for tagged frames will be offset by 4. This is based on Appendix-E of the IEEE 1588-2008 standard and the message format defined in **Table 30-4**.

Table 30-6 IPv6-UDP PTP Frame Fields Required for Control and Status

Field Matched	Octet Position	Matched Value	Description
MAC Frame type	12, 13	0x86DD	IP datagram
IP version	14 (bits [7:4])	0x6	IP version is IPv6
Layer-4 protocol	20[1)]	0x11	UDP
PTP Multicast address	38 – 53	FF0x:0:0:0:0:0:0:181 (Hex) FF02:0:0:0:0:0:0:6B (Hex)	PTP – primary multicast address: FF0x:0:0:0:0:0:0:0:181 (Hex) PTP – Pdelay multicast address: FF02:0:0:0:0:0:0:0:6B (Hex)

Table 30-6 IPv6-UDP PTP Frame Fields Required for Control and Status (cont'd)

Field Matched	Octet Position	Matched Value	Description
UDP destination port	56, 57 (*)	0x013F, 0x140	0x013F – PTP event message 0x0140 – PTP general messages
PTP control field (IEEE 1588 Version 1)	93 (*)	0x00/0x01/0x02/ 0x03/0x04	0x00 – SYNC, 0x01 – Delay_Req, 0x02 – Follow_Up 0x03 – Delay_Resp 0x04 – Management (version1)
PTP Message Type Field (IEEE version 2)	74 (*) (nibble)	0x0/0x1/0x2/0x3/ 0x8/0x9/0xB/0xC /0xD	0x0 – SYNC 0x1 – Delay_Req 0x2 – Pdelay_Req 0x3 – Pdelay_Resp 0x8 – Follow_Up 0x9 – Delay_Resp 0xA – Pdelay_Resp_Follow_Up 0xB – Announce 0xC – Signaling 0xD - Management
PTP version field	75 (nibble)	0x1 or 0x2	0x1 – Supports PTP version 1 0x2 – Supports PTP version 2

1) The Extension Header is not defined for PTP packets.

PTP Frame Over Ethernet

Table 30-7 gives the details of the fields that will be matched to control snapshot for PTP packets over Ethernet for IEEE 1588 version 1 and 2. Note that the octet positions for tagged frames will be offset by 4. This is based on Appendix-E of the IEEE 1588-2008 standard and the message format defined in **Table 30-4**.

Table 30-7 Ethernet PTP Frame Fields Required for Control And Status

Field Matched	Octet Position	Matched Value	Description
MAC Frame type	12, 13	0x88F7	PTP Ethernet frame.
PTP control field (IEEE Version 1)	45	0x00/0x01/0x02/ 0x03/0x04	0x00 – SYNC 0x01 – Delay_Req 0x02 – Follow_Up 0x03 – Delay_Resp 0x04 – Management
PTP Message Type Field (IEEE version 2)	14 (nibble)	0x0/0x1/0x2/0x3/0x 8/0x9/0xB/ 0xC/0xD	0x0 – SYNC 0x1 – Delay_Req 0x2 – Pdelay_Req 0x3 – Pdelay_Resp 0x8 – Follow_Up 0x9 – Delay_Resp 0xA – Pdelay_Resp_Follow_Up 0xB – Announce 0xC – Signaling 0xD – Management
MAC Destination multicast address[1)]	0-5	01-1B-19-00-00-00 01-80-C2-00-00-0E	All except peer delay messages - 01-1B-19-00-00-00 Pdelay messages - 01-80-C2-00-00-0E
PTP version field	15 (nibble)	0x1 or 0x2	0x1 – Supports PTP version 1 0x2 – Supports PTP version 2

1) In addition, the address match of destination addresses (DA) programmed in MAC address 1 to 31 will be used, if the control bit 18 (TSENMACADDR: Enable MAC address for PTP frame filtering) of the Time Stamp Control register is set.

Reference Timing Source (for Advance Timestamp Feature)

The updated functionality for advanced timestamp support is mentioned in the following points.

1. The IEEE 1588-2008 standard defines the seconds field of the time to be 48 bits wide. The fields to time-stamp will be the following.
   a) UInteger48- seconds field
   b) UInteger32-nanoseconds field
   The “seconds” field is the integer portion of the timestamp in units of seconds. The

"nanoseconds" field is the fractional portion of the timestamp in units of nanoseconds. E.g. 2.000000001 seconds is represented as secondsField = 0x0000_0000_0002 and nanoSeconds = 0x0000_0001. Thus the maximum value in nanoseconds field in this format will be 0x3B9A_C9FF value (i.e (10e9-1) nano-seconds). This is defined as digital rollover mode of operation. It will also support the older mode in which the nano-seconds field will roll-over and increment the seconds field after the value of 0x7FFF_FFFF. (Accuracy is ~0.466 ns per bit). This is defined as the binary rollover mode. The modes can be controlled using the "TSCTRLSSR: Time Stamp Digital or Binary rollover control" bit of the time stamp control register.

2. When the Advanced IEEE 1588 time-stamp feature is selected time maintained in the core will still be 64-bit wide, as the overflow to the upper 16-bits of seconds register happens once in 130 years. The value of the upper 16-bits of the seconds field can only be obtained from the CSR register. This is optional and can be controlled by the coreKit parameter "IEEE1588 Higher Word Register Enable" described in Chapter 6. A CSR status bit which indicates if the 32-bit "seconds" field has overflowed is also provided.
3. There is also a pulse-per-second output given to indicate 1 second interval (default). Option is provided to change the interval. Refer **32-bit Register - Timestamp_Control**. for more information.
4. Option to take auxiliary time-stamp snapshot with an external event is supported. Refer to section **Auxiliary Snapshot** for details.

Transmit Path Functions

There are no changes in the transmit path functions for GMAC-CORE and GMAC-MTL configuration for the Advanced time stamp option.

The structure of the descriptor changes when Advanced IEEE 1588 version support is enabled. The IEEE 1588 timestamp feature is supported using Alternate (Enhanced) descriptors format only. The descriptor is 32-bytes long (8 DWORDS) and the snapshot of the timestamp is written in descriptor 6 and 7.

Receive Path Functions

When the advanced time stamp feature is selected, processing of the received frames to identify valid PTP frames is done. The snapshot of the time to be sent to the application can be controlled.

The following options are provided in the time stamp control register to control the snapshot.

1. Option to enable snapshot for all frames.
2. Enable snapshot for IEEE 1588 version 2 or version 1 time stamp.
3. Enable snapshot for PTP frames transmitted directly over Ethernet or UDP-IP-Ethernet.

4. Enable time stamp snapshot for the received frame for IPv4 or IPv6.
5. Enable time stamp snapshot for EVENT messages (SYNC, DELAY_REQ, PDELAY_REQ or PDELAY_RESP) only.
6. Enable the node to be a Master or Slave. This will control the type of messages for which snap-shot will be taken (this depends on the type of clock that is selected and is valid for ordinary or boundary clock only).

Note that PTP messages over VLAN frames are also supported.

The time stamp is returned to the software inside the corresponding Transmit and Receive Descriptor. The Advanced time stamp feature is supported using Alternate (Enhanced) descriptors only which is 32-bytes long. Extended status containing the time stamp message status and the IPC offload status is written back in descriptor 4 and the snapshot of the time stamp is written back in descriptor 6 and 7. The complete details on the descriptors when the advanced time stamping is enabled are provided in **Alternate or Enhanced Descriptors**.

Auxiliary Snapshot

The auxiliary snap shot feature allows you to store a snapshot of the system time based on an external event. The event is considered to be the rising edge of the sideband signal ptp_aux_ts_trig_i. This feature is independent of whether the system time is generated internally or given as input (on ptp_timestamp_i bus). Such snap-shots are stored in a 4-deep FIFO. Only 64-bit of the time will be stored in the FIFO. The upper 16-bits of the seconds register can be read from “Time Stamp Higher Word Register” when it is present.

The storing of the snapshot can be indicated to the host with an interrupt. The value of the snapshot is read out through a FIFO register access (see **32-bit Register - Timestamp_Control**). When the FIFO becomes full and an external trigger to take the snapshot is asserted, then a snapshot trigger-missed status is set in the Time Stamp Status Register. This indicates that the latest auxiliary snapshot of the timestamp was not stored in the FIFO. The latest snapshot will not be written to the FIFO when it is full. When the host reads the 64-bit timestamp from the FIFO, space is available to store the next snapshot. This FIFO can be cleared using the control bit “ATSFC: Auxiliary Snapshot FIFO clear” defined in the Time Stamp Control Register. When multiple snapshots are present in the FIFO, the count is indicated in the “ATSNS: Auxiliary Time Stamp Number of Snapshots” bits of the Time Stamp Status Register.

30.2.3 AHB Application Host Interface

In the GMAC-AHB core, the DMA Controller interfaces with the Host through the AMBA AHB Interface. The AHB Master Interface controls data transfers while the AHB Slave interface accesses CSR space. The DMA can be used in embedded applications where DMA is required to optimize data transfer between the GMAC and system memory.

The AHB Master interface converts the internal DMA request cycles into AHB cycles.

Characteristics of this interface include the following:

- Fully AMBA 2.0-compliant AHB Master: with restrictions
- You can choose fixed burst length only (SINGLE, INCR4, INCR8) by programming the FB in the DMA Bus Mode register (see **Register Description**).
  – When fixed burst length is chosen, the AHB master always initiates a burst with SINGLE or INCR4/8type. But when such a burst is responded with SPLIT/RETRY/early burst termination, the AHB master will re-initiate the pending transfers of the burst with INCR or SINGLE burst-length type. It will terminate such INCR bursts when the original requested fixed-burst is transferred. In Fixed Burst-Length mode, if the DMA requests a burst transfer that is not equal to INCR4/8, the AHB Host interface splits the transfer into multiple burst transactions. For example, if the DMA requests a 15-beat burst transfer, the AHB interface splits it into multiple transfers of INCR8 and INCR4 and 3 SINGLE transactions.
- The AHB slaves interfaced to the GMAC-AHB must support SINGLE and INCR transfers, at a minimum.
- Takes care of AHB SPLIT, RETRY, and ERROR conditions. Any ERROR response will halt all further transactions for that DMA, and indicate the error as fatal through the CSR and interrupt. The application must give a hard or soft reset to the module to restart the operation.
- Takes care of AHB 1K boundary breaking
- Handles all data transfers (according to the data bus width configuration), except for Descriptor Status Write accesses (which are always 32-bit). In any burst data transfer, the address bus value is always aligned to the data bus width and need not be aligned to the beat size.
  All AHB burst transfers can be aligned to an address value by enabling the DMA Bus Mode register's AAL bit. If both the FB and AAL bits are set to 1, the AHB interface and the DMA together ensure that all initiated beats are aligned to the address, completing the frame transfer in the minimum number of required beats.
  For example, in 32-bit data bus mode, if a data buffer transfer's start address is 0xF000_0008 and the DMA is configured for a maximum beat size of 8, the AHB transfers occur in the following sequence:
  – 2 SINGLE transfers at addresses 0xF000_0008 and 0xF000_000C
  – 1 INCR4 transfer at address 0xF000_0010
  – All subsequent beats are INCR8 transfers starting from address 0xF000_0020. For an address-aligned INCR8 transfer, the 5 least-significant bits of the address must be 0.
- The DMA Controller requests an AHB Burst Read transfer only when it can accept the received burst data completely. Data read from the AHB is always pushed into the DMA without any delay or BUSY cycles.
- The DMA requests an AHB Burst Write transfer only when it has the sufficient data to transfer the burst completely. The AHB interface always assumes that it has data available to push into the AHB bus. However, the DMA can prematurely indicate end-

of-valid data (due to the transfer of end-of-frame of an Ethernet frame) during the burst. In Fixed Burst Length mode, the AHB Master interface continues the burst with dummy data until the specified length is completed.

The AHB 32-bit Slave interface provides access to the DMA and GMAC CSR space. Characteristics of this interface include the following:

- Fully AMBA 2.0 Compliant AHB Slave—no restrictions
- Supports single and INCR4/8 transfers
- Supports busy and early terminations
- Supports 32-bit, 16-bit, and 8-bit write/read transfers to the CSR; 32-bit access to the CSR are recommended to avoid any SW synchronization problems.
- Generates OKAY only response; does not generate SPLIT, RETRY, or ERROR responses.

30.2.4 DMA Controller

The DMA has independent Transmit and Receive engines, and a CSR space. The Transmit Engine transfers data from system memory to the device port (MTL), while the Receive Engine transfers data from the device port to system memory. The controller utilizes descriptors to efficiently move data from source to destination with minimal Host CPU intervention. The DMA is designed for packet-oriented data transfers such as frames in Ethernet. The controller can be programmed to interrupt the Host CPU for situations such as Frame Transmit and Receive transfer completion, and other normal/error conditions.

The DMA and the Host driver communicate through two data structures:

- Control and Status registers (CSR)
- Descriptor lists and data buffers

Control and Status registers are described in detail in **Chapter 30.3.1.1**. Descriptors are described in detail in **Chapter 30.4**.

The DMA transfers data frames received by the core to the Receive Buffer in the Host memory, and Transmit data frames from the Transmit Buffer in the Host memory. Descriptors that reside in the Host memory act as pointers to these buffers.

There are two descriptor lists; one for reception, and one for transmission. The base address of each list is written into DMA Registers 3 and 4, respectively. A descriptor list is forward linked (either implicitly or explicitly). The last descriptor may point back to the first entry to create a ring structure. Explicit chaining of descriptors is accomplished by setting the second address chained in both Receive and Transmit descriptors (RDES1[24] and TDES1[24]). The descriptor lists resides in the Host physical memory address space. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used, physically addressed, rather than contiguous buffers in memory.

A data buffer resides in the Host physical memory space, and consists of an entire frame or part of a frame, but cannot exceed a single frame. Buffers contain only data, buffer

status is maintained in the descriptor. Data chaining refers to frames that span multiple data buffers. However, a single descriptor cannot span multiple frames. The DMA will skip to the next frame buffer when end-of-frame is detected. Data chaining can be enabled or disabled.

The descriptor ring and chain structure is shown in **Figure 30-6**.

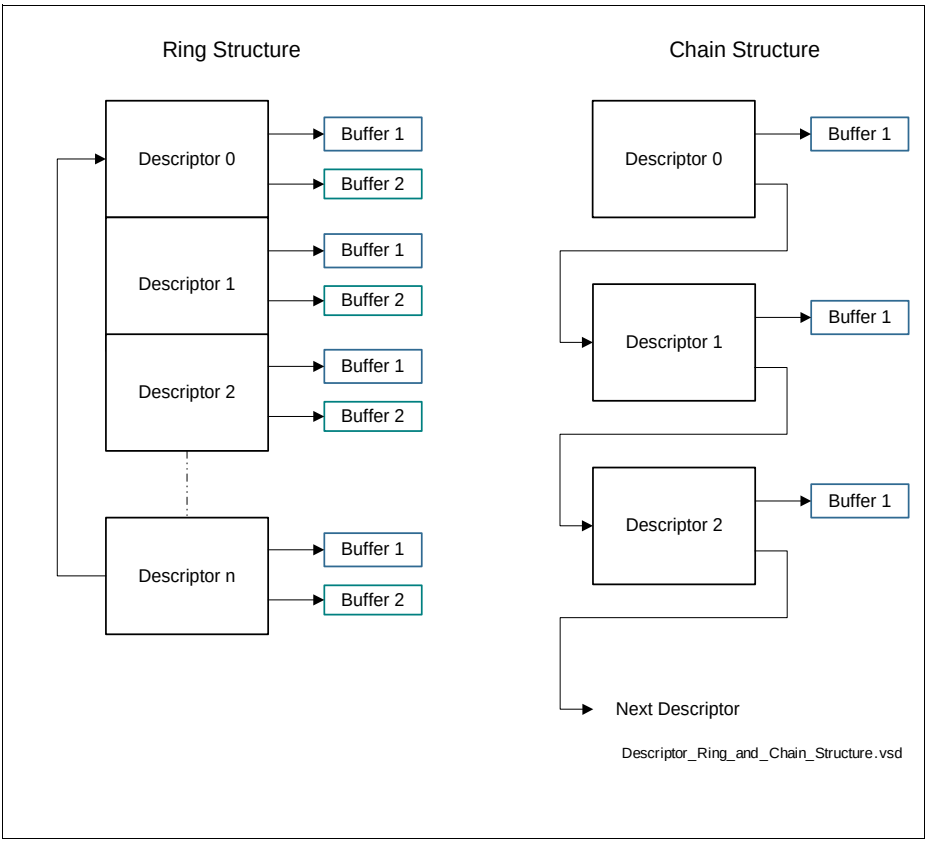


Figure 30-6 Descriptor Ring and Chain Structure

30.2.4.1 Initialization

Initialization for the GMAC is as follows.

1. Write to DMA Register 0 to set Host bus access parameters.
2. Write to DMA Register 7 to mask unnecessary interrupt causes.

3. The software driver creates the Transmit and Receive descriptor lists. Then it writes to both DMA Register 3 and DMA Register 4, providing the DMA with the starting address of each list.
4. Write to GMAC Registers 1, 2, and 3 for desired filtering options.
5. Write to GMAC Register 0 to configure and enable the Transmit and Receive operating modes. The PS and DM bits are set based on the auto-negotiation result (read from the PHY).
6. Write to DMA Register 6 to set bits 13 and 1 to start transmission and reception.
7. The Transmit and Receive engines enter the Running state and attempt to acquire descriptors from the respective descriptor lists. The Receive and Transmit engines then begin processing Receive and Transmit operations. The Transmit and Receive processes are independent of each other and can be started or stopped separately.

Host Bus Burst Access

The DMA will attempt to execute fixed-length Burst transfers on the AHBMaster interface if configured to do so (FB bit of DMA Register 0). The maximum Burst length is indicated and limited by the PBL field (DMA Register 0[13:8]). The Receive and Transmit descriptors are always accessed in the maximum possible (limited by PBL or 16 * 8/bus width) burst-size for the 16-bytes to be read.

The Transmit DMA will initiate a data transfer only when sufficient space to accommodate the configured burst is available in MTL Transmit FIFO or the number of bytes till the end of frame (when it is less than the configured burst-length). The DMA will indicate the start address and the number of transfers required to the AHB Master Interface. When the AHB Interface is configured for fixed-length burst, then it will transfer data using the best combination of INCR4/8 and SINGLE transactions.

The Receive DMA will initiate a data transfer only when sufficient data to accommodate the configured burst is available in MTL Receive FIFO or when the end of frame (when it is less than the configured burst-length) is detected in the Receive FIFO. The DMA will indicate the start address and the number of transfers required to the AHB Master Interface. When the AHB Interface is configured for fixed-length burst, then it will transfer data using the best combination of INCR4/8 and SINGLE transactions. If the end-of frame is reached before the fixed-burst ends on the AHB interface, then dummy transfers are performed in-order to complete the fixed-burst.

When the AHB interface is configured for address-aligned beats, both DMA engines ensure that the first burst transfer the AHB initiates is less than or equal to the size of the configured PBL. Thus, all subsequent beats start at an address that is aligned to the configured PBL. The DMA can only align the address for beats up to size 8 (for PBL > 8), because the AHB interface does not support more than INCR8.

Host Data Buffer Alignment

The Transmit and Receive data buffers do not have any restrictions on start address alignment. For example, in systems with 32-bit memory, the start address for the buffers can be aligned to any of the four bytes. However, the DMA always initiates transfers with address aligned to the bus width with dummy data for the byte lanes not required. This typically happens during the transfer of the beginning or end of an Ethernet frame.

Example - Buffer Read

If the Transmit buffer address is 32'h00000FF2 (for 32-bit data bus), and 15 bytes need to be transferred, then the DMA will read five full words from address 32'h00000FF0, but when transferring data to the MTL Transmit FIFO, the extra bytes (the first two bytes) will be dropped or ignored. Similarly, the last 3 bytes of the last transfer will also be ignored. The DMA always ensures it transfers a full 32-bit data to the MTL Transmit FIFO, unless it is the end-of-frame.

Buffer Size Calculations

The DMA does not update the size fields in the Transmit and Receive descriptors. The DMA updates only the status fields (RDES and TDES) of the descriptors. The driver has to perform the size calculations.

The transmit DMA transfers the exact number of bytes (indicated by buffer size field of TDES1) towards the GMAC core. If a descriptor is marked as first (FS bit of TDES1 is set), then the DMA marks the first transfer from the buffer as the start of frame. If a descriptor is marked as last (LS bit of TDES1), then the DMA marks the last transfer from that data buffer as the end-of frame to the MTL.

The Receive DMA transfers data to a buffer until the buffer is full or the end-of frame is received from the MTL. If a descriptor is not marked as last (LS bit of RDES0), then the descriptor's corresponding buffer(s) are full and the amount of valid data in a buffer is accurately indicated by its buffer size field minus the data buffer pointer offset when the FS bit of that descriptor is set. The offset is zero when the data buffer pointer is aligned to the data bus width. If a descriptor is marked as last, then the buffer may not be full (as indicated by the buffer size in RDES1). To compute the amount of valid data in this final buffer, the driver must read the frame length (FL bits of RDES0[29:16]) and subtract the sum of the buffer sizes of the preceding buffers in this frame. The Receive DMA always transfers the start of next frame with a new descriptor.

Note: Even when the start address of a receive buffer is not aligned to the system bus's data width, the system should allocate a receive buffer of a size aligned to the system bus width. For example, if the system allocates a 1.024-byte (1 KB) receive buffer starting from address 0x1000, the software can program the buffer start address in the Receive descriptor to have a 0x1002 offset. The Receive DMA writes the frame to this buffer with dummy data in the first two locations (0x1000 and 0x1001). The actual frame is written from location 0x1002. Thus, the actual

useful space in this buffer is 1.022 bytes, even though the buffer size is programmed as 1.024 bytes, due to the start address offset.

DMA Arbiter

The arbiter inside the DMA module performs the arbitration between the Transmit and Receive channel accesses to the AHB Master interface. Two types of arbitrations are possible: round-robin, and fixed-priority.

When round-robin arbitration is selected (DA bit of DMA Register 0 is reset), the arbiter allocates the data bus in the ratio set by the PR bits of DMA Register 0, when both Transmit and Receive DMAs are requesting for access simultaneously. When the DA bit is set, the Receive DMA always gets priority over the Transmit DMA for data access.

30.2.4.2 Transmission

TxDMA Operation: Default (Non-OSF) Mode

The Transmit DMA engine in default mode proceeds as follows:

1. The Host sets up the transmit descriptor (TDES0-TDES3) and sets the Own bit (TDES0[31]) after setting up the corresponding data buffer(s) with Ethernet Frame data.
2. Once the ST bit (DMA Register 6[13]) is set, the DMA enters the Run state.
3. While in the Run state, the DMA polls the Transmit Descriptor list for frames requiring transmission. After polling starts, it continues in either sequential descriptor ring order or chained order. If the DMA detects a descriptor flagged as owned by the Host, or if an error condition occurs, transmission is suspended and both the Transmit Buffer Unavailable (DMA Register 5[2]) and Normal Interrupt Summary (DMA Register 5[16]) bits are set. The Transmit Engine proceeds to Step 8.
4. If the acquired descriptor is flagged as owned by DMA (TDES0[31] = 1’b1), the DMA decodes the Transmit Data Buffer address from the acquired descriptor.
5. The DMA fetches the Transmit data from the Host memory and transfers the data to the MTL for transmission.
6. If an Ethernet frame is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Steps Step 2, Step 3, and Step 4 are repeated until the end-of-Ethernet-frame data is transferred to the MTL.
7. When frame transmission is complete, if IEEE 1588 time stamping was enabled for the frame (as indicated in the transmit status) the time-stamp value obtained from MTL is written to the transmit descriptor (TDES2 and TDES3) that contains the end-of-frame buffer. The status information is then written to this transmit descriptor (TDES0). Because the Own bit is cleared during this step, the Host now owns this

descriptor. If time stamping was not enabled for this frame, the DMA does not alter the contents of TDES2 and TDES3.

8. Transmit Interrupt (DMA Register 5[0]) is set after completing transmission of a frame that has Interrupt on Completion (TDES1[31]) set in its Last Descriptor. The DMA engine then returns to Step 2.
9. In the Suspend state, the DMA tries to re-acquire the descriptor (and thereby return to Step 2) when it receives a Transmit Poll demand and the Underflow Interrupt Status bit is cleared.

The TxDMA transmission flow in default mode is shown in **Figure 30-7**.

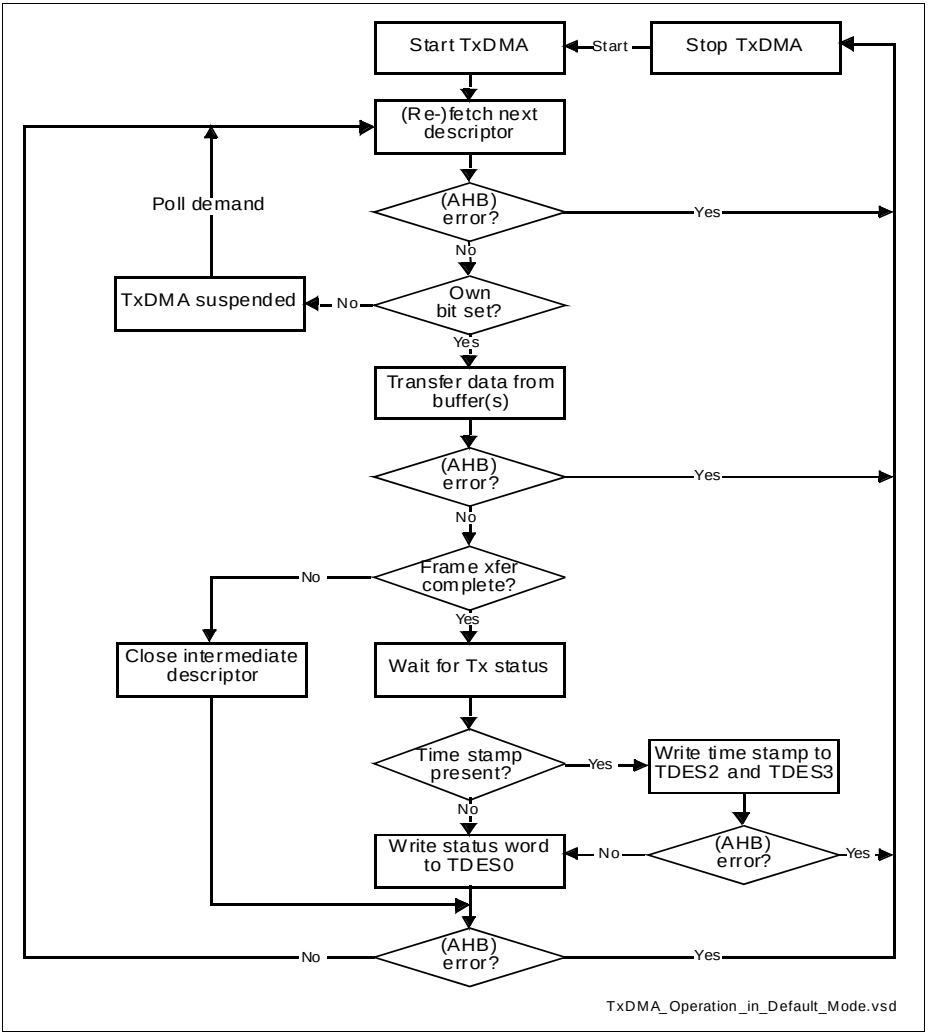


Figure 30-7 TxDMA Operation in Default Mode

TxDMA Operation: OSF Mode

While in the Run state, the transmit process can simultaneously acquire two frames without closing the Status descriptor of the first (if the OSF bit is set in DMA Register 6[2]). As the transmit process finishes transferring the first frame, it immediately polls the

Transmit Descriptor list for the second frame. If the second frame is valid, the transmit process transfers this frame before writing the first frame's status information.

In OSF mode, the Run state Transmit DMA operates in the following sequence:

1. The DMA operates as described in Step 1–Step 5 of the TxDMA (default mode).
2. Without closing the previous frame's last descriptor, the DMA fetches the next descriptor.
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into Suspend mode and skips to Step 6.
4. The DMA fetches the Transmit frame from the Host memory and transfers the frame to the MTL until the End-of-Frame data is transferred, closing the intermediate descriptors if this frame is split across multiple descriptors.
5. The DMA waits for the previous frame's frame transmission status and time stamp. Once the status is available, the DMA writes the time stamp to TDES2 and TDES3, if such time stamp was captured (as indicated by a status bit). The DMA then writes the status, with a cleared Own bit, to the corresponding TDES0, thus closing the descriptor. If time stamping was not enabled for the previous frame, the DMA does not alter the contents of TDES2 and TDES3.
6. If enabled, the Transmit interrupt is set, the DMA fetches the next descriptor, then proceeds to Step 2 (when Status is normal). If the previous transmission status shows an underflow error, the DMA goes into Suspend mode (Step 6).
7. In Suspend mode, if a pending status and time stamp are received from the MTL, the DMA writes the time stamp (if enabled for the current frame) to TDES2 and TDES3, then writes the status to the corresponding TDES0. It then sets relevant interrupts and returns to Suspend mode.
8. The DMA can exit Suspend mode and enter the Run state (go to Step 1 or Step 2 depending on pending status) only after receiving a Transmit Poll demand (DMA Register 1).

Note: As the DMA fetches the next descriptor in advance before closing the current descriptor, the descriptor chain should have more than 2 different descriptors for correct and proper operation.

The basic flow is charted in **Figure 30-8**.

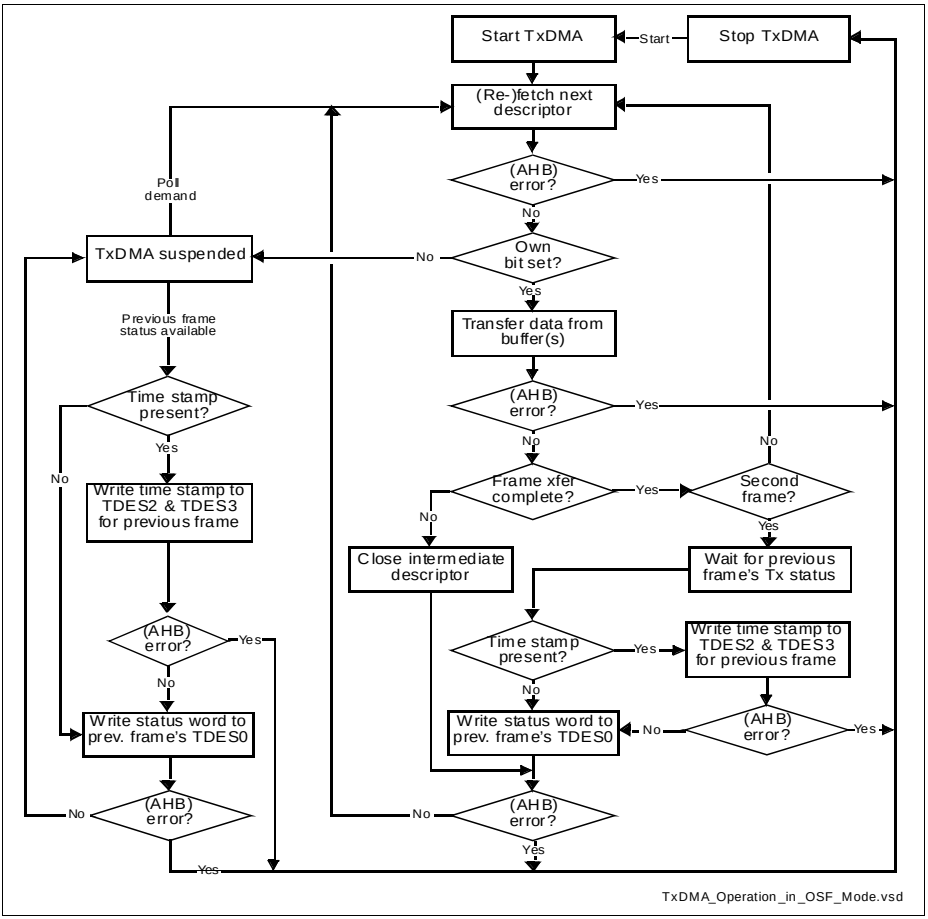


Figure 30-8 TxDMA Operation in OSF Mode

Transmit Frame Processing

The Transmit DMA expects that the data buffers contain complete Ethernet frames, excluding preamble, pad bytes, and FCS fields. The DA, SA, and Type/Len fields contain valid data. If the Transmit Descriptor indicates that the MAC core must disable CRC or PAD insertion, the buffer must have complete Ethernet frames (excluding preamble), including the CRC bytes.

Frames can be data-chained and can span several buffers. Frames must be delimited by the First Descriptor (TDES1[29]) and the Last Descriptor (TDES1[30]), respectively.

As transmission starts, the First Descriptor must have (TDES1[29]) set. When this occurs, frame data transfers from the Host buffer to the MTL Transmit FIFO. Concurrently, if the current frame has the Last Descriptor (TDES1[30]) clear, the Transmit Process attempts to acquire the Next Descriptor. The Transmit Process expects this descriptor to have TDES1[29] clear. If TDES1[30] is clear, it indicates an intermediary buffer. If TDES1[30] is set, it indicates the last buffer of the frame.

After the last buffer of the frame has been transmitted, the DMA writes back the final status information to the Transmit Descriptor 0 (TDES0) word of the descriptor that has the last segment set in Transmit Descriptor 1 (TDES1[30]). At this time, if Interrupt on Completion (TDES1[31]) was set, Transmit Interrupt (DMA Register 5[0]) is set, the Next Descriptor is fetched, and the process repeats.

Actual frame transmission begins after the MTL Transmit FIFO has reached either a programmable transmit threshold (DMA Register 6[16:14]), or a full frame is contained in the FIFO. There is also an option for Store and Forward Mode (DMA Register 6[21]). Descriptors are released (Own bit TDES0[31] clears) when the DMA finishes transferring the frame.

Transmit Polling Suspended

Transmit polling can be suspended by either of the following conditions:

- The DMA detects a descriptor owned by the Host (TDES0[31]=0). To resume, the driver must give descriptor ownership to the DMA and then issue a Poll Demand command.
- A frame transmission is aborted when a transmit error due to underflow is detected. The appropriate Transmit Descriptor 0 (TDES0) bit is set.

If the second condition occur, both Abnormal Interrupt Summary (DMA Register 5[15]) and Transmit Underflow bits (DMA Register 5 [5]) are set, and the information is written to Transmit Descriptor 0, causing the suspension. If the DMA goes into SUSPEND state due to the first condition, then both Normal Interrupt Summary (DMA Register 5 [16]) and Transmit Buffer Unavailable (DMA Register 5 [2]) are set.

In both cases, the position in the Transmit List is retained. The retained position is that of the descriptor following the Last Descriptor closed by the DMA.

The driver must explicitly issue a Transmit Poll Demand command after rectifying the suspension cause.

30.2.4.3 Reception

The Receive DMA engine's reception sequence is depicted in **Figure 30-9** and proceeds as follows:

1. The host sets up Receive descriptors (RDES0-RDES3) and sets the Own bit (RDES0[31]).

2. Once the SR (DMA Register 6[1]) bit is set, the DMA enters the Run state. While in the Run state, the DMA polls the Receive Descriptor list, attempting to acquire free descriptors. If the fetched descriptor is not free (is owned by the host), the DMA enters the Suspend state and jumps to Step 8.
3. The DMA decodes the receive data buffer address from the acquired descriptors.
4. Incoming frames are processed and placed in the acquired descriptor's data buffers.
5. When the buffer is full or the frame transfer is complete, the Receive engine fetches the next descriptor.
6. If the current frame transfer is complete, the DMA proceeds to Step 6. If the DMA does not own the next fetched descriptor and the frame transfer is not complete (EOF is not yet transferred), the DMA sets the Descriptor Error bit in the RDES0 (unless flushing is disabled). The DMA closes the current descriptor (clears the Own bit) and marks it as intermediate by clearing the Last Segment (LS) bit in the RDES0 value (marks it as Last Descriptor if flushing is not disabled), then proceeds to Step 7. If the DMA does own the next descriptor but the current frame transfer is not complete, the DMA closes the current descriptor as intermediate and reverts to Step 3.
7. If IEEE 1588 time stamping is enabled, the DMA writes the time stamp (if available) to the current descriptor's RDES2 and RDES3. It then takes the receive frame's status from the MTL and writes the status word to the current descriptor's RDES0, with the Own bit cleared and the Last Segment bit set.
8. The Receive engine checks the latest descriptor's Own bit. If the host owns the descriptor (Own bit is 1'b0) the Receive Buffer Unavailable bit (Register 5[7]) is set and the DMA Receive engine enters the Suspended state (Step 8). If the DMA owns the descriptor, the engine returns to Step 3 and awaits the next frame.
9. Before the Receive engine enters the Suspend state, partial frames are flushed from the Receive FIFO (You can control flushing using Bit 24 of DMA Register 6).
10. The Receive DMA exits the Suspend state when a Receive Poll demand is given or the start of next frame is available from the MTL's Receive FIFO. The engine proceeds to Step 1 and refetches the next descriptor.

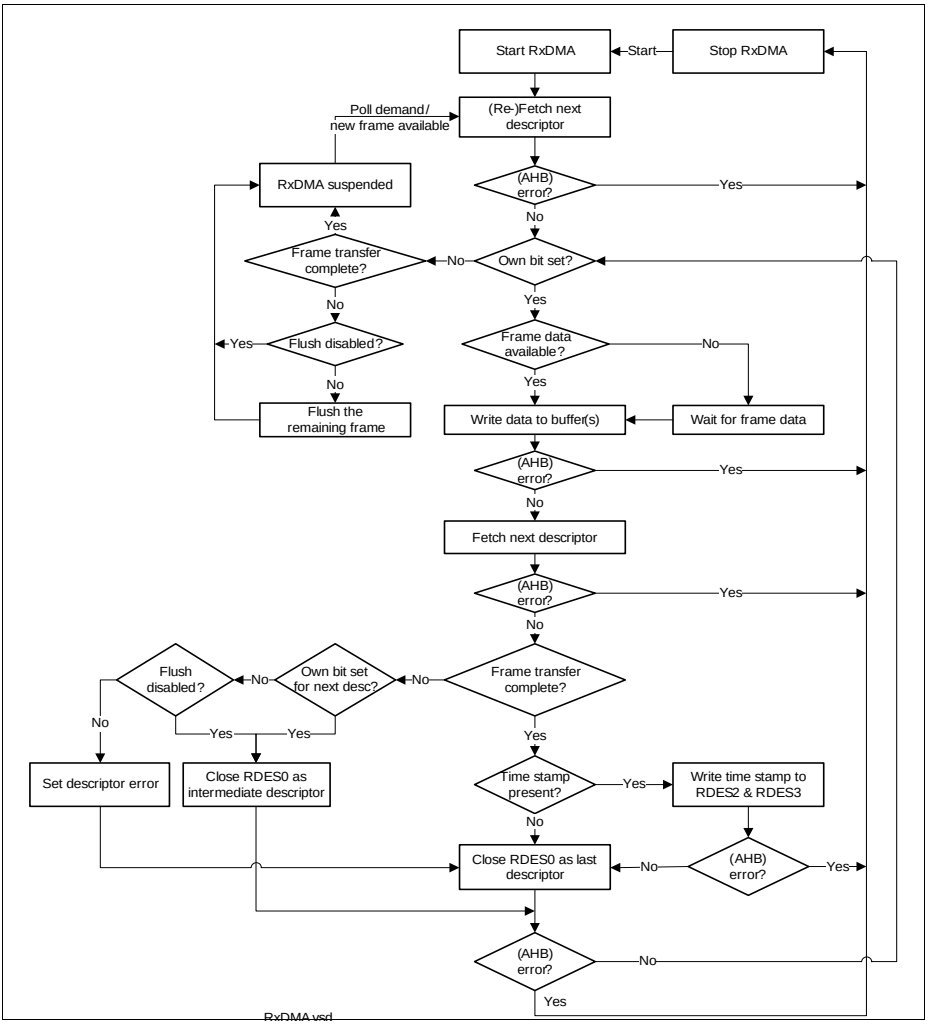


Figure 30-9 Receive DMA Operation

The DMA does not acknowledge accepting the status from the MTL until it has completed the time stamp write-back and is ready to perform status write-back to the descriptor.

If software has enabled time stamping through CSR, when a valid time stamp value is not available for the frame (for example, because the receive FIFO was full before the time stamp could be written to it), the DMA writes all-ones to RDES2 and RDES3.

Otherwise (that is, if time stamping is not enabled), the RDES2 and RDES3 remain unchanged.

Receive Descriptor Acquisition

The Receive Engine always attempts to acquire an extra descriptor in anticipation of an incoming frame. Descriptor acquisition is attempted if any of the following conditions is satisfied:

- The receive Start/Stop bit (Register 6[1]) has been set immediately after being placed in the Run state.
- The data buffer of current descriptor is full before the frame ends for the current transfer.
- The controller has completed frame reception, but the current Receive Descriptor is not yet closed.
- The receive process has been suspended because of a host-owned buffer (RDES0[31] = 0) and a new frame is received.
- A Receive poll demand has been issued.

Receive Frame Processing

The GMAC transfers the received frames to the Host memory only when the frame passes the address filter and frame size is greater than or equal to configurable threshold bytes set for the Receive FIFO of MTL, or when the complete frame is written to the FIFO in Store-and-Forward mode.

If the frame fails the address filtering, it is dropped in the GMAC block itself (unless Receive All GMAC Register 1[31] bit is set). Frames that are shorter than 64 bytes, because of collision or premature termination, can be purged from the MTL Receive FIFO.

After 64 (configurable threshold) bytes have been received, the MTL block requests the DMA block to begin transferring the frame data to the Receive Buffer pointed to by the current descriptor. The DMA sets First Descriptor (RDES0[9]) after the DMA Host Interface (AHB/AXI or MDC) becomes ready to receive a data transfer (if DMA is not fetching transmit data from the host), to delimit the frame. The descriptors are released when the Own (RDES[31]) bit is reset to 1’b0, either as the Data buffer fills up or as the last segment of the frame is transferred to the Receive buffer. If the frame is contained in a single descriptor, both Last Descriptor (RDES[8]) and First Descriptor (RDES[9]) are set.

The DMA fetches the next descriptor, sets the Last Descriptor (RDES[8]) bit, and releases the RDES0 status bits in the previous frame descriptor. Then the DMA sets Receive Interrupt (Register 5[6]). The same process repeats unless the DMA encounters a descriptor flagged as being owned by the host. If this occurs, the Receive Process sets Receive Buffer Unavailable (Register 5[7]) and then enters the Suspend state. The position in the receive list is retained.

Receive Process Suspended

If a new Receive frame arrives while the Receive Process is in Suspend state, the DMA refetches the current descriptor in the Host memory. If the descriptor is now owned by the DMA, the Receive Process re-enters the Run state and starts frame reception. If the descriptor is still owned by the host, by default, the DMA discards the current frame at the top of the MTL Rx FIFO and increments the missed frame counter. If more than one frame is stored in the MTL Rx FIFO, the process repeats.

The discarding or flushing of the frame at the top of the MTL Rx FIFO can be avoided by setting Operation Mode register bit 24 (DFF). In such conditions, the receive process sets the Receive Buffer Unavailable status and returns to the Suspend state.

30.2.4.4 Interrupts

Interrupts can be generated as a result of various events. DMA Register 5 contains all the bits that might cause an interrupt. DMA Register 7 contains an enable bit for each of the events that can cause an interrupt.

There are two groups of interrupts, Normal and Abnormal, as described in DMA Register 5. Interrupts are cleared by writing a 1’b1 to the corresponding bit position. When all the enabled interrupts within a group are cleared, the corresponding summary bit is cleared. When both the summary bits are cleared, the interrupt signal sbd_intr_o is deasserted. If the GMAC core is the cause for assertion of the interrupt, then any of the GLI, GMI, or GPI bits of DMA Register 5 will be set high.

Note: DMA Register 5 is the (interrupt) status register. The interrupt pin (sbd_intr_o) will be asserted due to any event in this status register only if the corresponding interrupt enable bit is set in DMA Register 7.

Interrupts are not queued and if the interrupt event occurs before the driver has responded to it, no additional interrupts are generated. For example, Receive Interrupt (DMA Register 5[6]) indicates that one or more frames was transferred to the Host buffer. The driver must scan all descriptors, from the last recorded position to the first one owned by the DMA.

An interrupt is generated only once for simultaneous, multiple events. The driver must scan DMA Register 5 for the cause of the interrupt. The interrupt is not generated again unless a new interrupting event occurs, after the driver has cleared the appropriate bit in DMA Register 5. For example, the controller generates a Receive interrupt (DMA Register 5[6]) and the driver begins reading DMA Register 5. Next, Receive Buffer Unavailable (DMA Register 5[7]) occurs. The driver clears the Receive interrupt. Even then, the sbd_intr_o signal is not deasserted, due to the active or pending Receive Buffer Unavailable interrupt.

An interrupt timer is given for flexible control of Receive Interrupt (DMA register 5[6]). When this Interrupt timer is programmed with a non-zero value, it will get activated as soon as the RxDMA completes a transfer of a received frame to system memory without

asserting the Receive Interrupt because it is not enabled in the corresponding Receive Descriptor (RDES1[31] in Table 7-3). When this timer runs out as per the programmed value, RI bit is set and the interrupt is asserted if the corresponding RI is enabled in DMA Register 7. This timer gets disabled before it runs out, when a frame is transferred to memory and the RI is set because it is enabled for that descriptor.

30.2.5 MAC Transaction Layer (MTL)

The MAC Transaction Layer provides FIFO memory to buffer and regulate the frames between the application system memory and the GMAC core. It also enables the data to be transferred between the application clock domain and the GMAC clock domains. The MTL layer has 2 data paths, namely the Transmit path and the Receive Path. The data path for both directions is 32-bit wide and operates with a simple FIFO protocol.

The GMAC-MTL communicates with the application side with the Application Transmit Interface (ATI), Application Receive Interface (ARI), and the MAC Control Interface (MCI).

30.2.5.1 Transmit Path

The DMA controls all transactions for the transmit path through the ATI. Ethernet frames read from the system memory is pushed into the FIFO by the DMA. The frame is then popped out and transferred to the GMAC core when triggered. When the end-of-frame is transferred, the status of the transmission is taken from the GMAC core and transferred back to the DMA.

The Transmit FIFO has a depth of 4K bytes. 4FIFO-fill level is indicated to the DMA so that it can initiate a data fetch in required bursts from the system memory, using the AHB interface. The data from the AHB Master interface is pushed into the FIFO with the appropriate byte lanes qualified by the DMA. The DMA also indicates the start-of-frame (SOF) and end-of-frame (EOF) transfers along with a few sideband signals controlling the pad-insertion/CRC generation for that frame in the GMAC core.

Per-frame control bits, such as Automatic Pad/CRC Stripping disable, time stamp capture, and so forth are taken as sideband control inputs on the ATI, stored in a separate register FIFO, and passed on to the core transmitter when the corresponding frame data is read from the Transmit FIFO.

There are two modes of operation for popping data towards the GMAC core. In Threshold mode, as soon as the number of bytes in the FIFO crosses the configured threshold level (or when the end-of-frame is written before the threshold is crossed), the data is ready to be popped out and forwarded to the GMAC core. The threshold level is configured using the TTC bits of DMA Register 0. In store-and-forward mode, the MTL pops the frame towards the GMAC core only when one or more of the following conditions are true:

- When a complete frame is stored in the FIFO

- When the TX FIFO becomes almost full
- When the ATI watermark becomes low. The watermark becomes low when the requested FIFO does not have space to accommodate the requested burst-length on the ATI.

Therefore, the MTL never stops in the store-and-forward mode even if the Ethernet frame length is bigger than the Tx FIFO depth.

The application can flush the Transmit FIFO of all contents by setting the FTF (DMA Register 6[20]) bit. This bit is self-clearing and initializes the FIFO pointers to the default state. If the FTF bit is set during a frame transfer from the MTL to the GMAC core, then the MTL stops further transfer as the FIFO is considered to be empty. Hence an underflow event occurs at the GMAC transmitter and the corresponding Status word is forwarded to the DMA.

Initialization through **Transmit Status Word** detail initialization and transmit operations for the MTL Layer.

Initialization

Upon reset, the MTL is ready to manage the flow of data to and from the DMA and the GMAC.

There are no requirements for enabling the MTL. However, the GMAC block and the DMA controller must be enabled individually through their respective CSRs.

Single-Packet Transmit Operation

During a transmit operation, the MTL block is slaved to the DMA controller. The general sequence of events for a transmit operation is as follows.

1. If the system has data to be transferred, the DMA controller, if enabled, fetches data from the Host through the AHB/AXI Master interface and starts forwarding it to the MTL. The MTL pushes the data received from the DMA into the FIFO. It continues to receive the data until the end-of frame of the frame is transferred.
2. The data is taken out of the FIFO and sent to the MAC by the FIFO controller engine. When the threshold level is crossed or a full packet of data is received into the FIFO, the MTL pops out the frame data and drives them to the GMAC core. The engine continues to transfer data from the FIFO until a complete packet has been transferred to the MAC. Upon completion of the frame, the MTL receives the Status from the GMAC and then notifies the DMA controller

Transmit Operation—Two Packets in the Buffer

1. Because the DMA must update the descriptor status before releasing it to the Host, there can be at the most two frames inside a transmit FIFO. The second frame will be fetched by the DMA and put into the FIFO only if the OSF (Operate on Second

Frame bit is set). If this bit is not set, the next frame will be fetched from the memory only after the MAC has completely processed the frame and the DMA has released the descriptors.

2. If the OSF bit is set, the DMA starts fetching the second frame immediately after completing the transfer of the first frame to the FIFO. It does not wait for the status to be updated. The MTL, in the meantime, receives the second frame into the FIFO while transmitting the first frame. As soon as the first frame has been transferred and the status is received from the MAC, the MTL pushes it to the DMA. If the DMA has already completed sending the second packet to the MTL, it must wait for the status of the first packet before proceeding to the next frame.

Transmit Operation—Multiple Packets in Buffer

In GMAC-MTL configuration, the transmit FIFO can be configured to accept more than 2 packets at a time. This option limits the number of status words that can be stored in the MTL before it is transferred to the DMA/host. By default, this number is limited to 2 but can be configured for 4 or 8 as well. Once the MTL FIFO accepts the number of frames equal to the status FIFO depth, it will stop accepting further frames unless the transmit Status that is given out and accepted by the host/DMA thus freeing up the space in this small FIFO.

Retransmission During Collision

While a frame is being transferred from the MTL to the GMAC, a collision event occurs on the GMAC line interface in Half-Duplex mode. The GMAC then indicates a retry attempt to the MTL by giving the status even before the end-of-frame is transferred from MTL. Then the MTL will enable the retransmission by popping out the frame again from the FIFO.

After more than 96 bytes are popped towards the GMAC core, the FIFO controller frees up that space and makes it available to the DMA to push in more data. This means that the retransmission is not possible after this threshold is crossed or when the GMAC core indicates a late-collision event.

Transmit FIFO Flush Operation

The GMAC provides a control to the software to flush the Transmit FIFO in the MTL layer through the use of Bit 20 of the Operation Mode register. The Flush operation is immediate and the MTL clears the Tx FIFO and the corresponding pointers to the initial state even if it is in the middle of transferring a frame to the GMAC Core. The data which is already accepted by the MAC transmitter will not be flushed. It will be scheduled for transmission and will result in underflow as TxFIFO does not complete the transfer of rest of the frame. As in all underflow conditions, a runt frame will be transmitted and observed on the line. The status of such a frame will be marked with both Underflow and Frame Flush events (TDES0 bits 13 and 1).

The MTL layer also stops accepting any data from the application (DMA) during the Flush operation. It will generate and transfer Transmit Status Words to the application for the number of frames that is flushed inside the MTL (including partial frames). Frames that are completely flushed in the MTL will have the Frame Flush Status bit (TDES0 13) set. The MTL completes the Flush operation when the application (DMA) accepts all of the Status Words for the frames that were flushed, and then clears the Transmit FIFO Flush control register bit. At this point, the MTL starts accepting new frames from the application (DMA).

Transmit Status Word

At the end of transfer of the Ethernet frame to the GMAC core and after the core completes the transmission of the frame, the MTL outputs the transmit status to the application.The detailed description of the Transmit Status is the same as for bits [23:0] of TDES0, given in **Table 30-28**.

If IEEE 1588 time stamping is enabled, the MTL returns specific frame's 64-bit time stamp, along with the ATI's transmit status.

Transmit Checksum Offload Engine

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. Because the most widespread use of Ethernet is to encapsulate TCP and UDP over IP datagrams, the GMAC-UNIV has an Checksum Offload Engine (COE) to support checksum calculation and insertion in the transmit path, and error detection in the receive path. This section explains the operation of the Checksum Offload Engine for transmitted frames.

Note: The checksum for TCP, UDP, or ICMP is calculated over a complete frame, then inserted into its corresponding header field. Due to this requirement, this function is enabled only when the Transmit FIFO is configured for Store-and-Forward mode (that is, when the TSF bit is set in DMA Register 6). If the core is configured for Threshold (cut-through) mode, the Transmit COE is bypassed.

Note: You must make sure that the Transmit FIFO is deep enough to store a complete frame before that frame is transferred to the GMAC Core transmitter. The reason being that when space is not available to accept the programmed burst length of the data, then the MTL TxFIFO starts reading to avoid dead-lock. Once reading starts, then checksum insertion engine fails and consequently all succeeding frames may get corrupted due to improper recovery. Therefore, you must enable the checksum insertion only in the frames that are less than the following number of bytes in size (even in the store-and-forward mode):
FIFO Depth – PBL – 3 FIFO Locations
The PBL is the programmed burst-length.

This module supports two types of checksum calculation and insertion. This checksum engine can be controlled for each frame by setting the CIC bits (Bits 28:27 of TDES1, described in **Transmit Descriptor 1 (TDES1)**) in GMAC-AHB or GMAC-DMA configurations.

Note: See IETF specifications RFC 791, RFC 793, RFC 768, RFC 792, RFC 2460, and RFC 4443 for IPv4, TCP, UDP, ICMP, IPv6, and ICMPv6 packet header specifications, respectively.

IP Header Checksum Engine

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit Header Checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the Ethernet frame's Type field has the value 0x0800 and the IP datagram's Version field has the value 0x4. The input frame's checksum field is ignored during calculation and replaced with the calculated value.

IPv6 headers do not have a checksum field; thus, the COE does not modify IPv6 header fields.

The result of this IP header checksum calculation is indicated by the IP Header Error status bit in the Transmit status (Bit 16 in **Table 30-28**). This status bit is set whenever the values of the Ethernet Type field and the IP header's Version field are not consistent, or when the Ethernet frame does not have enough data, as indicated by the IP header Length field.

In other words, this bit is set when an IP header error is asserted under the following circumstances:

For IPv4 datagrams

- The received Ethernet type is 0x0800, but the IP header's Version field does not equal 0x4
- The IPv4 Header Length field indicates a value less than 0x5 (20 bytes)
- The total frame length is less than the value given in the IPv4 Header Length field

For IPv6 datagrams

- The Ethernet type is 0x86dd but the IP header Version field does not equal 0x6
- The frame ends before the IPv6 header (40 bytes) or extension header (as given in the corresponding Header Length field in an extension header) is completely received.

Even when the COE detects such an IP header error, it inserts an IPv4 header checksum if the Ethernet Type field indicates an IPv4 payload.

TCP/UDP/ICMP Checksum Engine

The TCP/UDP/ICMP Checksum Engine processes the IPv4 or IPv6 header (including extension headers) and determines whether the encapsulated payload is TCP, UDP, or ICMP.

Note: For non-TCP, -UDP, or -ICMP/ICMPv6 payloads, this checksum engine is bypassed and nothing further is modified in the frame.

Note: Fragmented IP frames (IPv4 or IPv6), IP frames with security features (such as an authentication header or encapsulated security payload), and IPv6 frames with routing headers are not processed by this engine, and therefore must be bypassed. In other words, payload checksum insertion must not be enabled for such frames.

The checksum is calculated for the TCP, UDP, or ICMP payload and inserted into its corresponding field in the header. This engine can work in the following two modes:

- In the first mode, the TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the input frame's Checksum field. This engine includes the Checksum field in the checksum calculation, then replaces the Checksum field with the final calculated checksum.
- In the second mode, the engine ignores the Checksum field, includes the TCP, UDP, or ICMPv6 pseudo-header data into the checksum calculation, and overwrites the checksum field with the final calculated value.

Note: For ICMP-over-IPv4 packets, the Checksum field in the ICMP packet must always be 16'h0000 in both modes, because pseudo-headers are not defined for such packets. If it does not equal 16'h0000, an incorrect checksum may be inserted into the packet.

The result of this operation is indicated by the Payload Checksum Error status bit in the Transmit Status vector (Bit 12 in **Table 30-28**). This engine sets the Payload Checksum Error status bit when it detects that the frame has been forwarded to the MAC Transmitter engine in Store-and-Forward mode without the end-of-frame being written to the FIFO, or when the packet ends before the number of bytes indicated by the Payload Length field in the IP Header is received. When the packet is longer than the indicated payload length, the COE ignores them as stuff bytes, and no error is reported. When this engine detects the first type of error, it does not modify the TCP, UDP, or ICMP header. For the second error type, it still inserts the calculated checksum into the corresponding header field.

30.2.5.2 Receive Path

This module receives the frames given out by the GMAC core and pushes them into the Rx FIFO. The status (fill level) of this FIFO is indicated to the DMA once it crosses the configured Receive threshold (RTC of DMA Register 6). The MTL also indicates the FIFO fill level so that the DMA can initiate pre-configured burst transfers towards the AHB interface.

Receive Operation through **Receive Status Word** detail receive operations for the MTL Layer.

Receive Operation

During an Rx operation, the MTL is slaved to the GMAC. The general sequence of Receive operation events is as follows:

1. When the GMAC receives a frame, it pushes in data along with byte enables. The GMAC also indicates the SOF and EOF. The MTL accepts the data and pushes it into the Rx FIFO. After the EOF is transferred, the GMAC drives the status word, which is also pushed into the same Rx FIFO by the MTL.
2. When IEEE 1588 time stamping is enabled and the 64-bit time stamp is available along with the receive status, it is appended to the frame received from the GMAC and is pushed into the RxFIFO before the corresponding receive status word is written. Thus, two additional locations per frame are taken for storing the time stamp in the RxFIFO.
3. The MTL_RX engine takes the data out of the FIFO and sends it to the DMA. In the default Cut-Through mode, when 64 bytes (configured with RTC bits of DMA Register 6) or a full packet of data are received into the FIFO, the MTL_RX engine pops out the data and indicates its availability to the DMA. Once the DMA initiates the transfer to the AHBinterface, the MTL_RX engine continues to transfer data from the FIFO until a complete packet has been transferred. Upon completion of the EOF frame transfer, the MTL pops out the status word and sends it to the DMA controller.
4. In Rx FIFO Store-and-Forward mode (configured by the RSF bit of DMA Register 6), a frame is read out only after being written completely into the Receive FIFO. In this mode, all error frames are dropped (if the core is configured to do so) such that only valid frames are read out and forwarded to the application. In Cut-Through mode, some error frames are not dropped, because the error status is received at the end-of-frame, by which time the start of that frame has already been read out of the FIFO.

Note: In 32-bit data bus mode, the time-stamp transfer takes two clock cycles and the lower 32-bit of the time-stamp is given out first. The status also may be extended to two cycles when Advanced Time-stamp feature is enabled.

Receive Operation Multiframe Handling

Since the status is available immediately following the data, the MTL is capable of storing any number of frames into the FIFO, as long as it is not full.

GMAC Flow Control

The flow control operation of the GMAC can also be enabled (EFC bit of DMA Register 6) by the Rx FIFO control logic. The flow control signal to the GMAC is asserted whenever the Rx FIFO fill level crosses the configured threshold (RFA bits of DMA Register 6). This flow control signal is deasserted once the FIFO fill-level falls below the

configured threshold (RFD bits). This operation is independent of whether the GMAC is configured for full-duplex or half-duplex.

The above hardware flow control operation is applicable only when the Rx FIFO is 4.096 or more bytes deep. A separate sideband flow control signal (sbd_flowctrl_i) is optionally provided at the top-level I/O for all GMAC-AHB, GMAC-AXI, GMAC-DMA and GMAC-MTL configurations. For 4.096-byte or larger Rx FIFOs, this sideband signal is logically ORed with the hardware flow control signal. Thus, flow control is enabled when either of these signals is asserted and disabled when both these signals are deasserted.

Error Handling

If the MTL Rx FIFO is full before it receives the EOF data from the GMAC, an overflow is declared, the whole frame (including the status word) is dropped, and the overflow counter in the DMA (Register 8) is incremented. This is true even if the Forward Error Frame (FEF bit of DMA Register 6) is set. If the start address of such a frame has already been transferred to the Read Controller, the rest of the frame is dropped and a dummy EOF is written to the FIFO along with the status word. The status will indicate a partial frame due to overflow. In such frames, the Frame Length field is invalid.

The MTL Rx Control logic can filter error and undersized frames, if enabled (using the DMA Register 6 FEF and FUF bits). If the start address of such a frame has already been transferred to the Rx FIFO Read Controller, that frame is not filtered. The start address of the frame is transferred to the Read Controller after the frame crosses the receive threshold (set by the DMA Register 6 RTC bits).

If the MTL Receive FIFO is configured to operate in Store-and-Forward mode, all error frames can be filtered and dropped.

Receive Status Word

At the end of the transfer of the Ethernet frame to the host, the MTL outputs the receive status to the Application. The detailed description of the receive status is the same as for Bits[31:0] of RDES0, given in **Table 30-23**, except that Bits 31, 14, 9, and 8 are reserved and have a reset of 1'b0 by default. When the status of a partial frame due to overflow is given out, the Frame Length field in the status word is not valid.

*Note: When Advanced Time Stamp feature is enabled, the status is composed of two parts - normal (default [31:0]), and extended. The extended status[63:32] gives the information about the received ethernet payload when it is carrying PTP packets or TCP/UDP/ICMP over IP packets. In 32-bit data-bus, these are transferred over two clock cycles. The detailed description of the receive status is the same as described in RDES0 and RDES4 in **Receive Descriptor** , except that bits 31, 14, 9, and 8 of normal status is reserved and have a reset value of 1'b0. When the status of a partial frame due to overflow is given out, the Frame Length field in the status word is not valid.*

30.2.6 GMAC Core

The GMAC core supports many interfaces towards the PHY chip. In TC21x/TC22x/TC23x MII and RMII are implemented. The PHY interface can be selected only once after reset. The GMAC core communicates with the application side with the MAC Transmit Interface (MTI), MAC Receive Interface (MRI) and the MAC Control Interface (MCI).

30.2.6.1 Transmission

Transmission is initiated when the MTL Application pushes in data with the SOF . When the SOF signal is detected, the GMAC accepts the data and begins transmitting to the MII. The time required to transmit the frame data to the RMII/MII after the Application initiates transmission is variable, depending on delay factors like IFG delay, time to transmit preamble/SFD, and any back-off delays for Half-Duplex mode. Until then, the GMAC does not accept the data received from MTL.

After the EOF is transferred to the GMAC Core, the core complete normal transmission and then gives the Status of Transmission back to the MTL. If a normal collision (in Half-duplex mode) occurs during transmission, the GMAC core makes valid the Transmit Status to the MTL. It then accepts and drops all further data until the next SOF is received. The MTL block should retransmit the same frame from SOF on observing a Retry request (in the Status) from the GMAC.

The GMAC issues an underflow status if the MTL is not able to provide the data continuously during the transmission. During the normal transfer of a frame from MTL, if the GMAC receives a SOF without getting an EOF for the previous frame, then it (the SOF) is ignored and the new frame is considered as continuation of the previous frame.

The following six modules constitute the transmission function of the GMAC:

- Transmit Bus Interface Module (TBU)
- Transmit Frame Controller Module (TFC)
- Transmit Protocol Engine Module (TPE)
- Transmit Scheduler Module (STX)
- Transmit CRC Generator Module (CTX)
- Transmit Flow Control Module (FTX)

Transmit Bus Interface Module

This module interfaces the transmit path of the GMAC core with the external frame with a FIFO interface.

This module also outputs the (32-bit) Transmit Status to the application at the end of normal transmission or collision.

Additionally, this module outputs the Transmit Snapshot register value.

Transmit Frame Controller Module

The Transmit Frame Controller (TFC) consists of two registers to hold data, byte enables, and the last data control received from the TBU. The register provides a buffer between the Application and the TPE to regulate data flow as well as converts the input data into an 8-bit bus towards the TPE.

When the number of bytes received from the Application falls below 60 (DA+SA+LT+DATA), the state machine that interfaces with the TBU automatically appends zeros to the transmitting frame to make the data length exactly 46 bytes to meet the minimum data field requirement of IEEE 802.3. The GMAC can be programmed not to append any padding.

The cyclic redundancy check (CRC) for the Frame Check Sequence (FCS) field is calculated before transmission to the TPE module. This value is computed by CTX module. The TFC module receives the computed CRC and appends it to the data being transmitted to the TPE module. When the GMAC is programmed to not append the CRC value to the end of Ethernet frames, the TFC module ignores the computed CRC and transmits only the data received from the TBU module to the TPE module. An exception to this rule is that when the GMAC is programmed to append pads for frames (DA+SA+LT+DATA) less than 60 bytes sent by the TBU module, the TFC module will append the CRC at the end of padded frame.

The TFC converts the data received on the 32/64/128-bit interface from the TBU into 8-bit data for the TPE module.

Transmit Protocol Engine Module

The Transmit Protocol Engine (TPE) module consists of a transmit state machine that controls the operation of Ethernet frame transmission. The module's transmit state machine performs the following functions to meet the IEEE 802.3 specifications.

- Generates preamble and SFD
- Generates jam pattern in Half-Duplex mode
- Jabber timeout
- Flow control for Half-Duplex mode (back pressure)
- Generates transmit frame status
- Contains time stamp snapshot logic for IEEE 1588 support

When a new frame transmission from the TFC is requested, the transmit state machine sends out the preamble and SFD, followed by the data received. The preamble is defined as 7 bytes of 8'b10101010 pattern, and the SFD is defined as 1 byte of 8b'10101011 pattern.

The collision window is defined as 1 slot time (512 bit times for 10/100 Mbit/s Ethernet). The jam pattern generation is applicable only to Half-Duplex mode, not to Full-Duplex mode. In Full-Duplex mode, the transmit state machine ignores the phy_col_i signal from the PHY.

In MII mode, if a collision occurs any time from the beginning of the frame to the end of the CRC field, the transmit state machine sends a 32-bit jam pattern of 32'h55555555 on the MII to inform all other stations that a collision has occurred. If the collision is seen during the preamble transmission phase, the transmit state machine completes the transmission of preamble and SFD and then sends the jam pattern.

If the collision occurs after the collision window and before the end of the FCS field (or the end of Burst if the Frame Burst mode is enabled), the transmit state machine sends a 32-bit jam pattern and sets the late collision bit in the transmit frame status.

The TPE module maintains a jabber timer (only in 10/100-Mbit/sMbit/s mode) to cut off the transmission of Ethernet frames if the TFC module transfers more than 2.048 (default) bytes. The time-out is changed to 10.240 bytes when the Jumbo frame is enabled.

The Transmit state machine uses the deferral mechanism for the flow control (Back Pressure) in Half-Duplex mode. When the Application requests to stop receiving frames, the Transmit state machine sends a JAM pattern of 32 bytes whenever it senses a reception of a frame, provided the transmit flow control is enabled. This will result in a collision and the remote station will back off. The Application requests the flow control by setting BPA bit of Regsiter6. If the application requests a frame to be transmitted, then it will be scheduled and transmitted even when the backpressure is activated. Note that if the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur) then the remote stations will abort their transmissions due to excessive collisions.

If IEEE 1588 time stamping is enabled for the transmit frame, this block takes a snapshot of the system time when the SFD is put onto the transmit MII bus. The system time source is either an external input or internally generated, according to the configuration selected.

Transmit Scheduler Module

The Transmit Scheduler (STX) module is responsible for scheduling the frame transmission on the MII. The two major functions of this module are to maintain the inter-frame gap between two transmitted frames and to follow the Truncated Binary Exponential Back-off algorithm for Half-Duplex mode. This module provides an enable signal to the TPE module after satisfying the IFG and Back-off delays.

The STX module maintains an idle period of the configured inter-frame gap (IFG bits of Register 0 between any two transmitted frames. If frames from the TFC arrive at the TPE module sooner than the configured IFG time, the TPE module waits for the enable signal from the STX module before starting the transmission on the MII. The STX module starts its IFG counter as soon as the carrier signal of the MII goes inactive. At the end of programmed IFG value, the module issues an enable signal to the TPE module in Full-Duplex mode. In Half-Duplex mode and when IFG is configured for 96 bit times, the STX module follows the rule of deference specified in Section 4.2.3.2.1 of the IEEE 802.3

specification. The module resets its IFG counter if a carrier is detected during the first two-thirds (64-bit times for all IFG values) of the IFG interval. If the carrier is detected during the final one third of the IFG interval, the STX module continues the IFG count and enables the transmitter after the IFG interval.

The STX module implements the Truncated Binary Exponential Back-off algorithm when it operates in Half-Duplex mode.

Transmit CRC Generator Module

The Transmit CRC Generator (CTX) module interfaces with the TFC module to generate CRC for the FCS field of the Ethernet frame. The TFC module sends the frame data and any necessary padding to the CTX module through an 8-bit interface.

This module calculates the 32-bit CRC for the FCS field of the Ethernet frame. The encoding is defined by the following generating polynomial.

G (x) = x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1

The module gets the Ethernet frame's byte data from the TFC module (DA + SA + LT + DATA + PAD) qualified with a Data Valid signal. The TFC also indicates to the CTX when to reset the previously calculated CRC and to start the new CRC calculation for the coming frame. The TFC module issues the start command before sending the new frame data for calculation. The calculated CRC is valid on the next clock after the data is received.

Transmit Flow Control Module

The Transmit Flow Control (FTX) module generates Pause frames and transmit them to the TFC module as necessary, in Full-Duplex mode. The TFC module receives the Pause frame from the FTX module, appends the calculated CRC, and sends the frame to the TPE module. Pause frame generation can be initiated in two ways. The Application can request the FTX module to send a Pause frame either by setting the FCB bit in the Flow Control register Register 6 or by asserting the mti_flowctrl_i signal in response to the receive FIFO full conditions (packet buffer).

If the Application has requested the flow control by setting the FCB bit of Register 6, the FTX module will generate and transmit a single Pause frame to the TFC module. The value of the Pause Time in the generated frame contains the programmed Pause Time value in Register 6. To extend the pause or end the pause prior to the time specified in the previously transmitted Pause frame, the application must request another Pause frame transmission after programming the Pause Time register with appropriate value.

If the Application has requested the flow control by asserting the mti_flowctrl_i signal, the FTX module will generate and transmit a Pause frame to the TFC module. The value of the Pause Time in the generated frame contains the programmed Pause Time value in the Register 6. The FTX module monitors the mti_flowctrl_i signal. If it remains asserted at a configurable number of slot-times (PLT bits of GMAC Register 6) before this Pause-

time runs-out, a second Pause frame will be transmitted to the TFC module. The process will be repeated as long as the mti_flowctrl signal remains asserted.

If the mti_flowctrl_i signal goes inactive prior to the sampling time, the FTX module will transmit a Pause frame with zero Pause Time to indicate to the remote end that the receive buffer is ready to receive new data frames.

30.2.6.2 MAC Transmit Interface Protocol

The MAC Transmit Interface (MTI) connects the application (MTL module in the GMAC) with the GMAC to provide the Ethernet data for transmission.

The application initiates the Ethernet frame transmission by writing the first data of the frame to the GMAC, provided the GMAC is ready to accept data . The Application can push-in data as long as the GMAC core is ready to accept it.

If the frame transmission is not successful (due to underflow, collision, jabber timeout, excessive deferral events), the GMAC core will assert the transmit status even before the EOF is received. The Application will have to take the appropriate action as per the status. The GMAC will drop all further data input to it until the next SOF.

30.2.6.3 Reception

A receive operation is initiated when the GMAC detects an SFD on the MII. The core strips the preamble and SFD before proceeding to process the frame. The header fields are checked for the filtering and the FCS field used to verify the CRC for the frame. The received frame is stored in a shallow buffer until the address filtering is performed. The frame is dropped in the core if it fails the address filter.

The following are the functional blocks in the Receive path of the GMAC core.

- Receive Protocol Engine Module (RPE)
- Receive CRC Module (CRX)
- Receive Frame Controller Module (RFC)
- Receive Flow Control Module (FRX)
- Receive IP Checksum checker (IPC)
- Receive Bus Interface Unit Module (RBU)
- Address Filtering Module (AFM)

Receive Protocol Engine Module

The RPE consists of the receive state machine which strips the preamble and SFD. Once the phy_rxdv_i signal of the MII becomes active, the RPE's receive state machine begins hunting for the SFD field from the receive modifier logic. Until then, the state machine drops the receiving preambles. Once the SFD is detected, the state machine begins sending the data of the Ethernet frame to the RFC module, beginning with the first byte following the SFD (destination address).

If IEEE 1588 time stamping is enabled, the RPE takes a snapshot of the system time when any frame's SFD is detected on the MII. Unless the MAC filters out and drops the frame, this time stamp is passed on to the application.

In MII mode, the RPE converts the received nibble data into bytes, then forwards the valid frame data to the RFC module

The receive state machine of the RPE module decodes the Length/Type field of the receiving Ethernet frame. If the Length/Type field is less than 600 (hex) and if the MAC is programmed for the auto crc/pad stripping option, the state machine sends the data of the frame up to the count specified in the Length/Type field, then starts dropping bytes (including the FCS field). The state machine of the RPE module decodes the Length/Type field and checks for the Length interpretation.

If the Length/Type field is greater than or equal to 600 (hex), the RPE module will send all received Ethernet frame data to the RFC module, irrespective of the value on the programmed auto-CRC strip option.

As a default, the GMAC is programmed for watchdog timer to be enabled, that is, frames above 2.048 (10.240 if Jumbo Frame is enabled) bytes (DA + SA + LT + DATA + PAD + FCS) are cut off at the RPE module. This feature can be disabled by programming the GMAC Configuration register, Watchdog Disable. However even if the watchdog timer is disabled, frames greater than 16 KB in size are cut off and a watchdog time-out status is given.

The GMAC supports loopback of transmitted frames onto its receiver. As a default, the GMAC loopback function is disabled, but this feature can be enabled by programming the GMAC Configuration register, Loopback bit. The transmit and receive clocks can have an asynchronous timing relationship, so an asynchronous FIFO is used to make the loopback path of the phy_txd_o data onto the receive path. The asynchronous FIFO is 10 bits (6 bits in 10/100 Mbit/s mode) wide to accommodate phy_txd_o, phy_txen_o, and phy_txer_o. The FIFO nine deep in 10/100 Mbit/s mode and free-running to write on the write clock (clk_tx_i) and read on every read clock (clk_rx_i).

The write and read pointers gets re-initialized to have an offset of 2 (4 in 10/100 Mbit/s mode) at the start of each frame read out of the FIFO. This helps to avoid overflow/underflow during the transfer of a frame, and ensures that the overflow/underflow occurs only during the IFG period between the frames. Please note that the FIFO depth of nine is sufficient to prevent data corruption for frame sizes up to 9.022 bytes with a difference of 200 ppm between the MII Transmit and Receive clock frequencies. Hence, bigger frames should not be looped back, as they may get corrupted in this loopback FIFO.

At the end of every received frame, the RPE module generates received frame status and sends it to the RFC module. Control, missed frame, and filter fail status are added to the receive status in the RFC module.

Receive CRC Module

The Receive CRC (CRX) interfaces to the RPE module to check for any CRC error in the receiving frame.

This module calculates the 32-bit CRC for the received frame that includes the Destination address field through the FCS field. The encoding is defined by the following generating polynomial.

G (x) = x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1

The module gets the data from the RPE module (DA+SA+LT+DATA+PAD+FCS). The RPE module also sends a control signal that indicates the validity of the data. Irrespective of the auto pad/CRC strip, the CRX module receives the entire frame to compute the CRC check for received frame. As a note on the auto pad/CRC strip settings, the entire frame is not transferred between the RPE and RFC 8-bit interface.

Receive Checksum Offload Engine

The Checksum Offload engine is optional (not available in the default configuration) and selectable during configuration. Two types of Receive Checksum Offload engine are available for configuration. Full Checksum Offload, the Type 2 engine is instantiated.

Type 2

In this mode, both IPv4 and IPv6 frames in the received Ethernet frames are detected and processed for data integrity. You can enable this module by setting the IPC bit in the GMAC Configuration register. The GMAC receiver identifies IPv4 or IPv6 frames by checking for value 0x0800 or 0x86DD, respectively, in the received Ethernet frames' Type field. This identification applies to VLAN-tagged frames as well.

The Receive Checksum Offload engine calculates IPv4 header checksums and checks that they match the received IPv4 header checksums. The result of this operation (pass or fail) is given to the RFC module for insertion into the receive status word. The IP Header Error bit is set for any mismatch between the indicated payload type (Ethernet Type field) and the IP header version, or when the received frame does not have enough bytes, as indicated by the IPv4 header's Length field (or when fewer than 20 bytes are available in an IPv4 or IPv6 header).

This engine also identifies a TCP, UDP or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP/UDP/ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a Payload Checksum Error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not tally to the expected payload length given in the IP header.

As mentioned in **TCP/UDP/ICMP Checksum Engine**, this engine bypasses the payload of fragmented IP datagrams, IP datagrams with security features, IPv6 routing headers, and payloads other than TCP, UDP or ICMP.

In this configuration, the core does not append any payload checksum bytes to the received Ethernet frames.

Receive Frame Controller Module

The Receive Frame Controller (RFC) receives the Ethernet frame data and status from the RPE module. The RFC module consists of a FIFO of parameterized depth (default set to 4 deep and 37 bits wide) and two state machines for writing and reading the FIFO. The FIFO holds the received Ethernet frame data and byte enables, along with a control bit to indicate the last data. The state machines manage the FIFO and provide a frame buffering for the receiving Ethernet frame from the RPE module. The main functions of the RFC module are:

- Data path conversion, which converts the 8-bit data to 32-bit data to the RBU module.
- Frame filtering
- Attaching the calculated IP Checksum input from IPC.
- Update the Receive Status and forward to RBU.

If the RA bit of the GMAC CSR Frame Filter register is set, the RFC module initiates the data transfer to the RBU module as soon as 4 bytes of Ethernet data are received from the RPE module. At the end of the data transfer, the RFC module sends out the received frame status that includes the frame filter bits (SA Filterfail and DAFilterfail) and status from the RFC module. These bits are generated based on the filter-fail signals from the AFM module. This status bit indicates to the Application whether the received frame has passed the filter controls (both address filter and Frame Filter controls from CSR). The RFC module will not drop any frame on its own in this mode.

If the RA bit is reset, the RFC module performs frame filtering based on the destination/source address (the Application still needs to perform another level of filtering if it decides not to receive any bad frames like runt, CRC error frames, etc. The RFC module waits to receive the first 14 bytes of received data (type field) from the RPE module. Until then, the module will not initiate any transfers to the RBU module. After receiving the destination/source address bytes, the RFC checks the filter-fail signal from the AFM module for an address match. On detecting a filter-fail from AFB, the frame is dropped at the RFC module and not transferred to the Application.

On a delayed filter response from the AFM (this can only occur if you change the AFM logic), the RFC module waits until the FIFO is full, and then proceeds with the frame transfer to the RBU module. However, it will still take the delayed response from the AFM module and if it is a (DA/SA) filter failure, then it will drop the rest of the frame and send the Rx Status Word (with zero frame-length, CRC Error and Runt Error bits set) immediately indicating the filter-fail. If there is no response from the AFM until the end of frame is transmitted, the filter fail status in the Rx Status Word is updated accordingly.

When the PMT module is configured for power-down mode, all received frames are dropped by this block, and are not forwarded to the application.

Receive Flow Control Module

The Receive Flow Controller (FRX) detects the receiving Pause frame and pauses the frame transmission for the delay specified within the received Pause frame. The FRX module is enabled only in Full-Duplex mode. The Pause frame detection function can be enabled or disabled with the RFE bit of GMAC CSR Register 6.

Once the receive flow control is enabled, the FRX module begins monitoring the received frame destination address for any match with the multicast address of the control frame (48'h0180C2000001). If a match is detected, the FRX module indicates to the RFC module, that the destination address of the received frame matches the reserved control frame destination address. The RFC module then decides whether or not to transfer the received control frame to the Application, based on the (PCF) bit setting of GMAC CSR Register 1 (Filter register).

The FRX module also decodes the Type, Op-code, and Pause Timer field of the receiving control frame. At the end of received frame, the FRX module gets the received frame status from RPE. If the byte count of the status indicates 64 bytes, and if there is no CRC error, the FRX module requests the MAC transmitter to pause the transmission of any data frame for the duration of the decoded Pause Time value, multiplied by the slot time (64 byte times). Meanwhile, if another Pause frame is detected with a zero Pause Time value, the FRX module resets the Pause Time and gives another pause request to the Transmitter. If the received control frame matches neither the Type field (16'h8808), Opcode (16'h00001), nor byte length (64 bytes), or if there is a CRC error, the FRX module does not generate a Pause request to Transmitter.

In the case of a pause frame with a multicast destination address, the RFC filters the frame based on the address match from the FRX module. For a pause frame with a unicast destination address, the filtering in the FRX module depends on whether the DA matched the contents of the MAC Address Register 0 and the UP Bit of GMAC Core Register 6 is set (detecting a pause frame even with a unicast destination address). The PCF register bits (Bit [7:6] of GMAC Register 1) controls the filtering for control frames in addition to the Address filter module.

Receive Bus Interface Unit Module

The Receive Bus Interface Unit (RBU) converts the 32-bit data received from the RFC module into a 32-bit FIFO protocol on the Application side. The RBU module interfaces with the Application through the MAC receive interface (MRI).

If IEEE 1588 time stamping is enabled, the RBU also outputs the time stamp captured from the received frame.

Address Filtering Module

The Address Filtering (AFM) module performs the destination and source address checking function on all received frames and reports the address filtering status to the RFC module. The address checking is based on different parameters (Frame Filter register) chosen by the Application. These parameters are inputs to the AFM module as control signals, and the AFM module reports the status of the address filtering based on the combination of these inputs. The AFM module does not filter the receive frames by itself, but reports the status of the address filtering (whether to drop the frame or not) to the RFC module. The AFM module also reports whether the receiving frame is a multicast frame or a broadcast frame, as well as the address filter status.

The AFM module probes the 8-bit receive data path between the RPE module and the RFC module and checks the destination and source address field of each incoming packet. In MII mode the module takes 14/26 clocks (from the start of frame) to compare the destination/ source address of the receiving frame. The AFM module gets the station's physical (MAC) address and the Multicast Hash table from CSR module for address checking. The CSR module provides the Frame Filter register parameters to AFM.

Unicast Destination Address Filter

The AFM supports up to 32 MAC addresses for unicast perfect filtering. If perfect filtering is selected (HUC bit of Frame Filter register is reset), the AFM compares all 48 bits of the received unicast address with the programmed MAC address for any match. Default MacAddr0 is always enabled, other addresses MacAddr1–MacAddr31 are selected with an individual enable bit. Each byte of these other addresses (MacAddr1–MacAddr31) can be masked during comparison with the corresponding received DA byte by setting the corresponding Mask Byte Control bit in the register. This helps group address filtering for the DA.

In Hash filtering mode (When HUC bit is set), the AFM performs imperfect filtering for unicast addresses using a 64-bit Hash table. For hash filtering, the AFM uses the upper 6 bits CRC of the received destination address to index the content of the Hash table. A value of 000000 selects Bit 0 of the selected register, and a value of 111111 selects Bit 63 of the Hash Table register. If the corresponding bit (indicated by the 6-bit CRC) is set to 1, the unicast frame is said to have passed the Hash filter; otherwise, the frame has failed the Hash filter.

Multicast Destination Address Filter

The GMAC can be programmed to pass all multicast frames by setting the PM bit in the Frame Filter register. If the PM bit is reset, the AFM performs the filtering for multicast addresses based on the HMC bit of Frame Filter register. In Perfect Filtering mode, the multicast address is compared with the programmed MAC Destination Address registers (1–31). Group address filtering is also supported.

In Hash filtering mode, the AFM performs imperfect filtering using a 64-bit Hash table. For hash filtering, the AFM uses the upper 6 bits CRC of the received multicast address to index the content of the Hash table. A value of 000000 selects Bit 0 of the selected register and a value of 111111 selects Bit 63 of the Hash Table register.

If the corresponding bit is set to 1, then the multicast frame is said to have passed the Hash filter; otherwise, the frame has failed the Hash filter.

Hash or Perfect Address Filter

The DA filter can be configured to pass a frame when its DA matches either the Hash filter or the Perfect filter by setting the HPF bit of the Frame Filter register and setting the corresponding HUC or HMC bits. This configuration applies to both unicast and multicast frames. If the HPF bit is reset, only one of the filters (Hash or Perfect) is applied to the received frame.

Broadcast Address Filter

The AFM doesn't filter any broadcast frames in the default mode. However, if the GMAC is programmed to reject all broadcast frames by setting the DBF bit in the Frame Filter register, the DAF module asserts the Filter fail signal to RFC, whenever a broadcast frame is received. This will tell the RFC module to drop the frame.

Unicast Source Address Filter

The GMAC can also perform a perfect filtering based on the source address field of the received frames. By default, the AFM compares the SA field with the values programmed in the SA registers. The MAC Address registers [1:31] can be configured to contain SA instead of DA for comparison, by setting Bit 30 of the corresponding Register. Group filtering with SA is also supported. The frames that fail the SA Filter are dropped by the GMAC if the SAF bit of Frame Filter register is set.

When SAF bit is set, the result of SA Filter and DA filter is AND'ed to decide whether the frame needs to be forwarded. This means that either of the filter fail result will drop the frame and both filters have to pass in-order to forward the frame to the application.

Inverse Filtering Operation

For both Destination and Source address filtering, there is an option to invert the filter-match result at the final output. These are controlled by the DAIF and SAIF bits of the Frame Filter register respectively. The DAIF bit is applicable for both Unicast and Multicast DA frames. The result of the unicast/multicast destination address filter is inverted in this mode. Similarly, when the SAIF bit is set, the result of unicast SA filter is reversed.

Table 30-8 and **Table 30-9** summarize the Destination and Source Address filtering based on the type of frames received.

Table 30-8 Destination Address Filtering Table

Frame Type	PR	HPF	HUC	DAIF	HMC	PM	DB	DA Filter Operation
Broadcast	1	X	X	X	X	X	X	Pass
	0	X	X	X	X	X	0	Pass
	0	X	X	X	X	X	1	Fail
Unicast	1	X	X	X	X	X	X	Pass all frames.
	0	X	0	0	X	X	X	Pass on Perfect/Group filter match.
	0	X	0	1	X	X	X	Fail on Perfect/Group filter match.
	0	0	1	0	X	X	X	Pass on Hash filter match.
	0	0	1	1	X	X	X	Fail on Hash filter match.
	0	1	1	0	X	X	X	Pass on Hash or Perfect/Group filter match.
	0	1	1	1	X	X	X	Fail on Hash or Perfect/Group filter match.
Multicast	1	X	X	X	X	X	X	Pass all frames.
	X	X	X	X	X	1	X	Pass all frames.
	0	X	X	0	0	0	X	Pass on Perfect/Group filter match and drop PAUSE control frames if PCF = 0x.
	0	0	X	0	1	0	X	Pass on Hash filter match and drop PAUSE control frames if PCF = 0x.
	0	1	X	0	1	0	X	Pass on Hash or Perfect/Group filter match and drop PAUSE control frames if PCF = 0x.
	0	X	X	1	0	0	X	Fail on Perfect/Group filter match and drop PAUSE control frames if PCF = 0x.

Table 30-8 Destination Address Filtering Table (cont'd)

Frame Type	PR	HPF	HUC	DAIF	HMC	PM	DB	DA Filter Operation
	0	0	X	1	1	0	X	Fail on Hash filter match and drop PAUSE control frames if PCF = 0x.
	0	1	X	1	1	0	X	Fail on Hash or Perfect/Group filter match and drop PAUSE control frames if PCF = 0x.

Table 30-9 Source Address Filtering Table

Frame Type	PR	SAIF	SAF	SA Filter Operation
Unicast	1	X	X	Pass all frames.
	0	0	0	Pass status on Perfect/Group filter match but do not drop frames that fail.
	0	1	0	Fail status on Perfect/Group filter match but do not drop frame.
	0	0	1	Pass on Perfect/Group filter match and drop frames that fail.
	0	1	1	Fail on Perfect/Group filter match and drop frames that fail.

30.2.6.4 System Time Register Module

64-bit time is maintained in this module and updated using the input reference clock. This time is the source for taking snapshots (time stamps) of Ethernet frames being transmitted or received at the MII.

The System Time counter can be initialized or corrected using the coarse correction method. In this method, the initial value or the offset value is written to the Time Stamp Update register. For initialization, the System Time counter is written with the value in the Time Stamp Update registers, while for system time correction, the offset value is added to or subtracted from the system time.

In the fine correction method, a slave clock's frequency drift with respect to the master clock (as defined in IEEE 1588) is corrected over a period of time instead of in one clock, as in coarse correction. This helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP Sync message intervals. In this method, an accumulator sums up the contents of the Addend register, as shown in

Figure 30-10. The arithmetic carry that the accumulator generates is used as a pulse to increment the system time counter. The accumulator and the addend are 32-bit registers. Here, the accumulator acts as a high-precision frequency multiplier or divider. This algorithm is depicted in **Figure 30-10**:

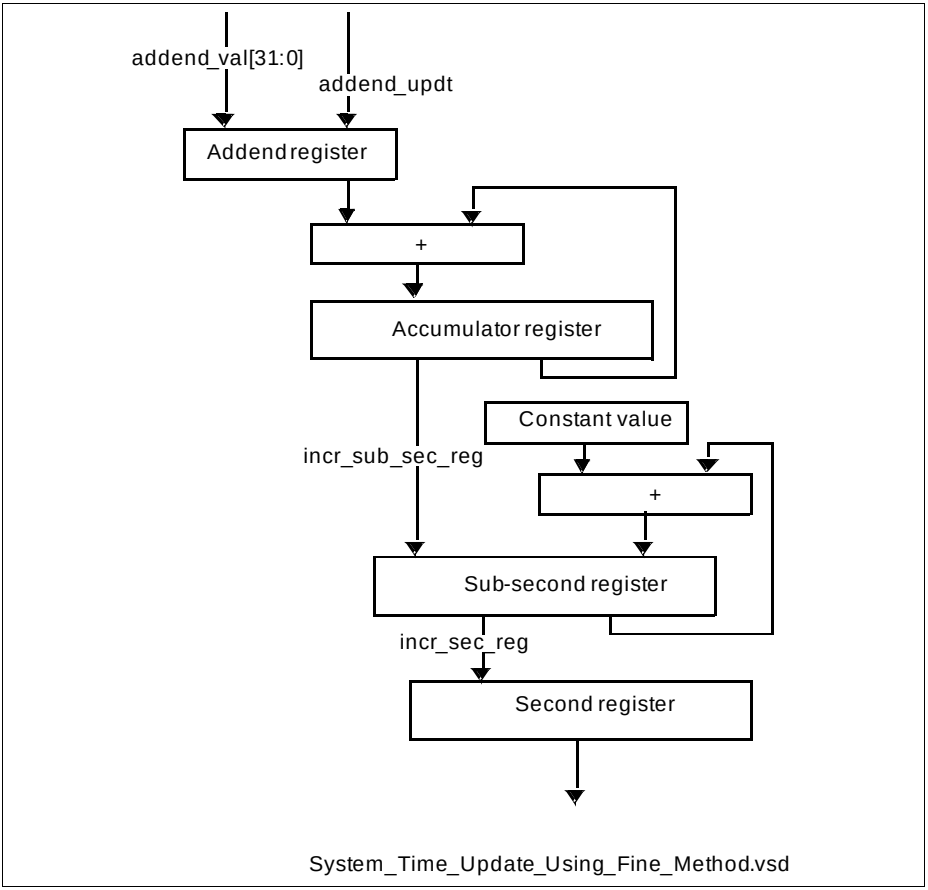


Figure 30-10 System Time Update Using Fine Method

The System Time Update logic requires a 50-MHz clock frequency to achieve 20-ns accuracy. The frequency division is the ratio of the reference clock frequency to the required clock frequency. Hence, if the reference clock (clk_ptp_ref_i) is, for example, 66 MHz, this ratio is calculated as 66 MHz / 50 MHz = 1.32. Hence, the default addend value to be set in the register is 2^{32} / 1.32, 0xC1F07C1F.

If the reference clock drifts lower, to 65 MHz for example, the ratio is 65 / 50, or 1.3 and the value to set in the addend register is 2^{32} / 1.30, or 0xC4EC4EC4. If the clock drifts higher, to 67 MHz for example, the addend register must be set to 0xBF0B7672. When the clock drift is nil, the default addend value of 0xC1F07C1F (2^{32} / 1.32) must be programmed.

In **Figure 30-10**, the constant value used to accumulate the sub-second register is decimal 43, which achieves an accuracy of 20 ns in the system time (in other words, it is incremented in 20-ns steps). Two different methods are used to update the System Time register, depending on which configuration you choose (See **Architecture**).

The software must calculate the drift in frequency based on the Sync messages and update the Addend register accordingly.

Initially, the slave clock is set with FreqCompensationValue0 in the Addend register. This value is as follows:

$FreqCompensationValue_0 = 2^{32} / FreqDivisionRatio$

If MasterToSlaveDelay is initially assumed to be the same for consecutive Sync messages, the algorithm described below must be applied. After a few Sync cycles, frequency lock occurs. The slave clock can then determine a precise MasterToSlaveDelay value and re-synchronize with the master using the new value.

The algorithm is as follows:

- At time $MasterSyncTime_n$ the master sends the slave clock a Sync message. The slave receives this message when its local clock is $SlaveClockTime_n$ and computes $MasterClockTime_n$ as:
  $MasterClockTime_n = MasterSyncTime_n + MasterToSlaveDelay_n$
- The master clock count for current Sync cycle, $MasterClockCount_n$ is given by:
  $MasterClockCount_n = MasterClockTime_n – MasterClockTime_{n-1}$ (assuming that MasterToSlaveDelay is the same for Sync cycles n and n – 1)
- The slave clock count for current Sync cycle, $SlaveClockCount_n$ is given by:
  $SlaveClockCount_n = SlaveClockTime_n – SlaveClockTime_{n-1}$
- The difference between master and slave clock counts for current Sync cycle, $ClockDiffCount_n$ is given by:
  $ClockDiffCount_n = MasterClockCount_n – SlaveClockCount_n$
- The frequency-scaling factor for slave clock, $FreqScaleFactor_n$ is given by:
  $FreqScaleFactor_n = (MasterClockCount_n + ClockDiffCount_n) / SlaveClockCount_n$
- The frequency compensation value for Addend register, $FreqCompensationValue_n$ is given by:
  $FreqCompensationValue_n = FreqScaleFactor_n * FreqCompensationValue_{n-1}$

In theory, this algorithm achieves lock in one Sync cycle; however, it may take several cycles, due to changing network propagation delays and operating conditions.

This algorithm is self-correcting: if for any reason the slave clock is initially set to a value from the master that is incorrect, the algorithm will correct it at the cost of more Sync cycles.

30.2.7 MAC Management Counters

The counters in the MAC Management Counters (MMC) module can be viewed as an extension of the register address space of the CSR module. The MMC module maintains a set of registers for gathering statistics on the received and transmitted frames. These include a control register for controlling the behavior of the registers, two 32-bit registers containing interrupts generated (receive and transmit), and two 32-bit registers containing masks for the Interrupt register (receive and transmit). These registers are accessible from the Application through the MAC Control Interface (MCI). Each register is 32 bits wide. The write data is qualified with the corresponding mci_be_i signals. Thus, non-32-bit accesses are allowed as long as the address is word-aligned.

The organization of these registers is shown in **Table 1-17**. The MMCs are accessed using transactions, in the same way the CSR address space is accessed. The following sections in the chapter describe the various counters and list the address for each of the statistics counters. This address will be used for Read/Write accesses to the desired transmit/receive counter.

The Receive MMC counters are updated for frames that are passed by the Address Filter (AFM) block. Statistics of frames that are dropped by the AFM module are not updated unless they are runt frames of less than 6 bytes (DA bytes are not received fully).

The MMC module gathers statistics on encapsulated IPv4, IPv6, TCP, UDP, or ICMP payloads in received Ethernet frames. This gathering is only enabled when Full Checksum Offload Engine is selected during RTL configuration. The address map of the corresponding registers, 0x0200–0x02FC, is given in **Table 1-17**.

30.2.7.1 Address Assignments

The MMC register naming convention is as follows.

- “tx” as a prefix or suffix indicates counters associated with transmission
- “rx” as a prefix or suffix indicates counters associated with reception
- “_g” as a suffix indicates registers that count good frames only
- “_gb” as a suffix indicates registers that count frames regardless of whether they are good or bad

The following explanations pertain to terminology used in **Table 1-17** through **Table 1-24**.

Transmitted frames are considered “good” if transmitted successfully. In other words, a transmitted frame is good if the frame transmission is not aborted due to any of the following errors:

- Jabber Timeout

- No Carrier/Loss of Carrier
- Late Collision
- Frame Underflow
- Excessive Deferral
- Excessive Collision

Received frames are considered “good” if none of the following errors exists:

- CRC error
- Runt Frame (shorter than 64 bytes)
- Alignment error (in 10/100 Mbit/s only)
- Length error (non-Type frames only)
- Out of Range (non-Type frames only, longer than maximum size)

The maximum frame size depends on the frame type, as follows:

- Untagged frame maxsize = 1518
- VLAN Frame maxsize = 1522
- Jumbo Frame maxsize = 9018
- JumboVLAN Frame maxsize = 9022

Table 30-10 MMC Register Map

GMAC CSR Register No.	Address Offset	Register Name	Register Description
64	0x0100	mmc_cntrl	MMC Control establishes the operating mode of MMC. For more details, refer to **Table 1-18**.
65	0x0104	mmc_intr_rx	MMC Receive Interrupt maintains the interrupt generated from all of the receive statistic counters. For more details, refer to **Table 1-19**.
66	0x0108	mmc_intr_tx	MMC Transmit Interrupt maintains the interrupt generated from all of the transmit statistic counters. For more details, refer to **Table 1-20**.
67	0x010C	mmc_intr_mask_rx	MMC Receive Interrupt mask maintains the mask for the interrupt generated from all of the receive statistic counters. For more details, refer to **Table 1-21**.

Table 30-10 MMC Register Map (cont'd)

GMAC CSR Register No.	Address Offset	Register Name	Register Description
68	0x0110	mmc_intr_mask_tx	MMC Transmit Interrupt Mask maintains the mask for the interrupt generated from all of the transmit statistic counters. For more details, refer to **Table 1-22**.
69	0x0114	txoctetcount_gb	Number of bytes transmitted, exclusive of preamble and retried bytes, in good and bad frames.
70	0x0118	txframecount_gb	Number of good and bad frames transmitted, exclusive of retried frames.
71	0x011C	txbroadcastframes_g	Number of good broadcast frames transmitted.
72	0x0120	txmulticastframes_g	Number of good multicast frames transmitted.
73	0x0124	tx64octets_gb	Number of good and bad frames transmitted with length 64 bytes, exclusive of preamble and retried frames.
74	0x0128	tx65to127octets_gb	Number of good and bad frames transmitted with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried frames.
75	0x012C	tx128to255octets_gb	Number of good and bad frames transmitted with length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried frames.
76	0x0130	tx256to511octets_gb	Number of good and bad frames transmitted with length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried frames.
77	0x0134	tx512to1023octets_gb	Number of good and bad frames transmitted with length between 512 and 1.023 (inclusive) bytes, exclusive of preamble and retried frames.

Table 30-10 MMC Register Map (cont'd)

GMAC CSR Register No.	Address Offset	Register Name	Register Description
78	0x0138	tx1024tomaxoctets_gb	Number of good and bad frames transmitted with length between 1.024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames.
79	0x013C	txunicastframes_gb	Number of good and bad unicast frames transmitted.
80	0x0140	txmulticastframes_gb	Number of good and bad multicast frames transmitted.
81	0x0144	txbroadcastframes_gb	Number of good and bad broadcast frames transmitted.
82	0x0148	txunderflowerror	Number of frames aborted due to frame underflow error.
83	0x014C	txsinglecol_g	Number of successfully transmitted frames after a single collision in Half-duplex mode.
84	0x0150	txmulticol_g	Number of successfully transmitted frames after more than a single collision in Half-duplex mode.
85	0x0154	txdeferred	Number of successfully transmitted frames after a deferral in Half-duplex mode.
86	0x0158	txlatecol	Number of frames aborted due to late collision error.
87	0x015C	txexesscol	Number of frames aborted due to excessive (16) collision errors.
88	0x0160	txcarriererror	Number of frames aborted due to carrier sense error (no carrier or loss of carrier).
89	0x0164	txoctetcount_g	Number of bytes transmitted, exclusive of preamble, in good frames only.
90	0x0168	txframecount_g	Number of good frames transmitted.
91	0x016C	txexcessdef	Number of frames aborted due to excessive deferral error (deferred for more than two max-sized frame times).

Table 30-10 MMC Register Map (cont'd)

GMAC CSR Register No.	Address Offset	Register Name	Register Description
92	0x0170	txpauseframes	Number of good PAUSE frames transmitted.
93	0x0174	txvlanframes_g	Number of good VLAN frames transmitted, exclusive of retried frames.
94	0x0178	Reserved	
95	0x017C	Reserved	
96	0x0180	rxframecount_gb	Number of good and bad frames received.
97	0x0184	rxoctetcount_gb	Number of bytes received, exclusive of preamble, in good and bad frames.
98	0x0188	rxoctetcount_g	Number of bytes received, exclusive of preamble, only in good frames.
99	0x018C	rxbroadcastframes_g	Number of good broadcast frames received.
100	0x0190	rxmulticastframes_g	Number of good multicast frames received.
101	0x0194	rxcrcerror	Number of frames received with CRC error.
102	0x0198	rxalignmenterror	Number of frames received with alignment (dribble) error.
103	0x019C	rxrunterror	Number of frames received with runt (<64 bytes and CRC error) error.
104	0x01A0	rxjabbererror	Number of giant frames received with length (including CRC) greater than 1.518 bytes (1.522 bytes for VLAN tagged) and with CRC error. If Jumbo Frame mode is enabled, then frames of length greater than 9,018 bytes (9,022 for VLAN tagged) are considered as giant frames.
105	0x01A4	rxundersize_g	Number of frames received with length less than 64 bytes, without any errors.
106	0x01A8	rxoversize_g	Number of frames received with length greater than the maxsize (1.518 or 1.522 for VLAN tagged frames), without errors.

Table 30-10 MMC Register Map (cont'd)

GMAC CSR Register No.	Address Offset	Register Name	Register Description
107	0x01AC	rx64octets_gb	Number of good and bad frames received with length 64 bytes, exclusive of preamble.
108	0x01B0	rx65to127octets_gb	Number of good and bad frames received with length between 65 and 127 (inclusive) bytes, exclusive of preamble.
109	0x01B4	rx128to255octets_gb	Number of good and bad frames received with length between 128 and 255 (inclusive) bytes, exclusive of preamble.
110	0x01B8	rx256to511octets_gb	Number of good and bad frames received with length between 256 and 511 (inclusive) bytes, exclusive of preamble.
111	0x01BC	rx512to1023octets_gb	Number of good and bad frames received with length between 512 and 1.023 (inclusive) bytes, exclusive of preamble.
112	0x01C0	rx1024tomaxoctets_gb	Number of good and bad frames received with length between 1.024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames.
113	0x01C4	rxunicastframes_g	Number of good unicast frames received.
114	0x01C8	rxlengtherror	Number of frames received with length error (Length type field ¼ frame size), for all frames with valid length field.
115	0x01CC	rxoutofrangetype	Number of frames received with length field not equal to the valid frame size (greater than 1.500 but less than 1.536).
116	0x01D0	rxpauseframes	Number of good and valid PAUSE frames received.
117	0x01D4	rxfifooverflow	Number of missed received frames due to FIFO overflow.
118	0x01D8	rxvlanframes_gb	Number of good and bad VLAN frames received.

Table 30-10 MMC Register Map (cont'd)

GMAC CSR Register No.	Address Offset	Register Name	Register Description
119	0x01DC	rxwatchdogerror	Number of frames received with error due to watchdog timeout error (frames with a data load larger than 2.048 bytes).
120:127	0x01E0–0x01FC	Reserved	
128	0x0200	mmc_ipc_intr_mask_rx	MMC IPC Receive Checksum Offload Interrupt Mask maintains the mask for the interrupt generated from the receive IPC statistic counters. See **Table 1-23** for further detail.
129	0x0204	Reserved	
130	0x0208	mmc_ipc_intr_rx	MMC Receive Checksum Offload Interrupt maintains the interrupt that the receive IPC statistic counters generate. See **Table 1-24** for further detail.
131	0x020C	Reserved	
132	0x0210	rxipv4_gd_frms	Number of good IPv4 datagrams received with the TCP, UDP, or ICMP payload
133	0x0214	rxipv4_hdrerr_frms	Number of IPv4 datagrams received with header (checksum, length, or version mismatch) errors
134	0x0218	rxipv4_nopay_frms	Number of IPv4 datagram frames received that did not have a TCP, UDP, or ICMP payload processed by the Checksum engine
135	0x021C	rxipv4_frag_frms	Number of good IPv4 datagrams with fragmentation
136	0x0220	rxipv4_udsbl_frms	Number of good IPv4 datagrams received that had a UDP payload with checksum disabled
137	0x0224	rxipv6_gd_frms	Number of good IPv6 datagrams received with TCP, UDP, or ICMP payloads

Table 30-10 MMC Register Map (cont'd)

GMAC CSR Register No.	Address Offset	Register Name	Register Description
138	0x0228	rxipv6_hdrerr_frms	Number of IPv6 datagrams received with header errors (length or version mismatch)
139	0x022C	rxipv6_nopay_frms	Number of IPv6 datagram frames received that did not have a TCP, UDP, or ICMP payload. This includes all IPv6 datagrams with fragmentation or security extension headers
140	0x0230	rxudp_gd_frms	Number of good IP datagrams with a good UDP payload. This counter is not updated when the rxipv4_udsbl_frms counter is incremented.
141	0x0234	rxudp_err_frms	Number of good IP datagrams whose UDP payload has a checksum error
142	0x0238	rxtcp_gd_frms	Number of good IP datagrams with a good TCP payload
143	0x023C	rxtcp_err_frms	Number of good IP datagrams whose TCP payload has a checksum error
144	0x0240	rxicmp_gd_frms	Number of good IP datagrams with a good ICMP payload
145	0x0244	rxicmp_err_frms	Number of good IP datagrams whose ICMP payload has a checksum error
146:147	0x0248–0x024C	Reserved	
148	0x0250	rxipv4_gd_octets	Number of bytes received in good IPv4 datagrams encapsulating TCP, UDP, or ICMP data. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter or in the octet counters listed below).
149	0x0254	rxipv4_hdrerr_octets	Number of bytes received in IPv4 datagrams with header errors (checksum, length, version mismatch). The value in the Length field of IPv4 header is used to update this counter.

Table 30-10 MMC Register Map (cont'd)

GMAC CSR Register No.	Address Offset	Register Name	Register Description
150	0x0258	rxipv4_nopay_octets	Number of bytes received in IPv4 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the IPv4 header's Length field is used to update this counter.
151	0x025C	rxipv4_frag_octets	Number of bytes received in fragmented IPv4 datagrams. The value in the IPv4 header's Length field is used to update this counter.
152	0x0260	rxipv4_udsbl_octets	Number of bytes received in a UDP segment that had the UDP checksum disabled. This counter does not count IP Header bytes.
153	0x0264	rxipv6_gd_octets	Number of bytes received in good IPv6 datagrams encapsulating TCP, UDP or ICMPv6 data
154	0x0268	rxipv6_hdrerr_octets	Number of bytes received in IPv6 datagrams with header errors (length, version mismatch). The value in the IPv6 header's Length field is used to update this counter.
155	0x026C	rxipv6_nopay_octets	Number of bytes received in IPv6 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the IPv6 header's Length field is used to update this counter.
156	0x0270	rxudp_gd_octets	Number of bytes received in a good UDP segment. This counter (and the counters below) does not count IP header bytes.
157	0x0274	rxudp_err_octets	Number of bytes received in a UDP segment that had checksum errors
158	0x0278	rxtcp_gd_octets	Number of bytes received in a good TCP segment

Table 30-10 MMC Register Map (cont'd)

GMAC CSR Register No.	Address Offset	Register Name	Register Description
159	0x027C	rxtcp_err_octets	Number of bytes received in a TCP segment with checksum errors
160	0x0280	rxicmp_gd_octets	Number of bytes received in a good ICMP segment
161	0x0284	rxicmp_err_octets	Number of bytes received in an ICMP segment with checksum errors
162:191	0x0288–0x02FC	Reserved	

30.2.7.2 MMC Register Description

MMC Control Register

The MMC Control register establishes the operating mode of the management counters.

Table 30-11 MMC Control Register

Bit	Access Type	Reset Value	Description
31:6	R	0000_000H	Reserved
5	rw	0	Full-Half preset When low and bit4 is set, all MMC counters get preset to almost-half value. All octet counters get preset to 0x7FFF_F800 (half - 2KBytes) and all frame-counters gets preset to 0x7FFF_FFF0 (half - 16) When high and bit4 is set, all MMC counters get preset to almost-full value. All octet counters get preset to 0xFFFF_F800 (full - 2KBytes) and all frame-counters gets preset to 0xFFFF_FFF0 (full - 16)

Table 30-11 MMC Control Register (cont'd)

Bit	Access Type	Reset Value	Description
4	rwh	0	Counters Preset When set, all counters will be initialized or preset to almost full or almost half as per Bit5 above. This bit will be cleared automatically after 1 clock cycle. This bit along with bit5 is useful for debugging and testing the assertion of interrupts due to MMC counter becoming half-full or full.
3	rw	0	MMC Counter Freeze When set, this bit freezes all the MMC counters to their current value. (None of the MMC counters are updated due to any transmitted or received frame until this bit is reset to 0. If any MMC counter is read with the Reset on Read bit set, then that counter is also cleared in this mode.)
2	rw	0	Reset on Read When set, the MMC counters will be reset to zero after Read (self-clearing after reset). The counters are cleared when the least significant byte lane (bits[7:0]) is read.
1	rw	0	Counter Stop Rollover When set, counter after reaching maximum value will not roll over to zero.
0	rwh	0	Counters Reset When set, all counters will be reset. This bit will be cleared automatically after 1 clock cycle

Note: When Bit 0 and Bit 4 are set in the same cycle, Counters get preset immediately after getting cleared.

MMC Receive Interrupt Register

The MMC Receive Interrupt register maintains the interrupts generated when the receive statistic counters reach half their maximum values (0x8000_0000), and when they cross their maximum values (0xFFFF_FFFF). When Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Receive Interrupt register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (bits[7:0]) of the respective counter must be read in order to clear the interrupt bit.

Note: Access restriction applies: Any read access to the register clears this register!

Table 30-12 MMC Receive Interrupt Register

Bit	Access Type	Reset Value	Description
31:24	r	00H	Reserved
23	rh	0	The bit is set when the rxwatchdog error counter reaches half the maximum value, and also when it reaches the maximum value.
22	rh	0	The bit is set when the rxvlanframes_gb counter reaches half the maximum value, and also when it reaches the maximum value.
21	rh	0	The bit is set when the rxfifooverflow counter reaches half the maximum value, and also when it reaches the maximum value.
20	rh	0	The bit is set when the rxpauseframes counter reaches half the maximum value, and also when it reaches the maximum value.
19	rh	0	The bit is set when the rxoutofrangetype counter reaches half the maximum value, and also when it reaches the maximum value.
18	rh	0	The bit is set when the rxlengtherror counter reaches half the maximum value, and also when it reaches the maximum value.
17	rh	0	The bit is set when the rxunicastframes_g counter reaches half the maximum value, and also when it reaches the maximum value.
16	rh	0	The bit is set when the rx1024tomaxoctets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
15	rh	0	The bit is set when the rx512to1023octets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
14	rh	0	The bit is set when the rx256to511octets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
13	rh	0	The bit is set when the rx128to255octets_gb counter reaches half the maximum value, and also when it reaches the maximum value.

Table 30-12 MMC Receive Interrupt Register (cont'd)

Bit	Access Type	Reset Value	Description
12	rh	0	The bit is set when the rx65to127octets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
11	rh	0	The bit is set when the rx64octets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
10	rh	0	The bit is set when the rxoversize_g counter reaches half the maximum value, and also when it reaches the maximum value.
9	rh	0	The bit is set when the rxundersize_g counter reaches half the maximum value, and also when it reaches the maximum value.
8	rh	0	The bit is set when the rxjabbererror counter reaches half the maximum value, and also when it reaches the maximum value.
7	rh	0	The bit is set when the rxrunterror counter reaches half the maximum value, and also when it reaches the maximum value.
6	rh	0	The bit is set when the rxalignmenterror counter reaches half the maximum value, and also when it reaches the maximum value.
5	rh	0	The bit is set when the rxcrcerror counter reaches half the maximum value, and also when it reaches the maximum value.
4	rh	0	The bit is set when the rxmulticastframes_g counter reaches half the maximum value, and also when it reaches the maximum value.
3	rh	0	The bit is set when the rxbroadcastframes_g counter reaches half the maximum value, and also when it reaches the maximum value.
2	rh	0	The bit is set when the rxoctetcount_g counter reaches half the maximum value, and also when it reaches the maximum value.

Table 30-12 MMC Receive Interrupt Register (cont'd)

Bit	Access Type	Reset Value	Description
1	rh	0	The bit is set when the rxoctetcount_gb counter reaches half the maximum value, and also when it reaches the maximum value.
0	rh	0	The bit is set when the rxframecount_gb counter reaches half the maximum value, and also when it reaches the maximum value.

Note: rh means that this register bit is set internally and is cleared when the Counter register is read.

MMC Transmit Interrupt Register

The MMC Transmit Interrupt register maintains the interrupts generated when transmit statistic counters reach half their maximum values (0x8000_0000), and when they cross their maximum values (0xFFFF_FFFF). When Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Transmit Interrupt register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (bits[7:0]) of the respective counter must be read in order to clear the interrupt bit.

Note: Access restriction applies: Any read access to the register clears this register!

Table 30-13 MMC Transmit Interrupt Register

Bit	Access Type	Reset Value	Description
31:25	r	00_H	Reserved
24	rh	0	The bit is set when the txvlanframes_g counter reaches half the maximum value, and also when it reaches the maximum value.
23	rh	0	The bit is set when the txpauseframes error counter reaches half the maximum value, and also when it reaches the maximum value.
22	rh	0	The bit is set when the txoexcessdef counter reaches half the maximum value, and also when it reaches the maximum value.
21	rh	0	The bit is set when the txframecount_g counter reaches half the maximum value, and also when it reaches the maximum value.

Table 30-13 MMC Transmit Interrupt Register (cont'd)

Bit	Access Type	Reset Value	Description
20	rh	0	The bit is set when the txoctetcount_g counter reaches half the maximum value, and also when it reaches the maximum value.
19	rh	0	The bit is set when the txcarriererror counter reaches half the maximum value, and also when it reaches the maximum value.
18	rh	0	The bit is set when the txexesscol counter reaches half the maximum value, and also when it reaches the maximum value.
17	rh	0	The bit is set when the txlatecol counter reaches half the maximum value, and also when it reaches the maximum value.
16	rh	0	The bit is set when the txdeferred counter reaches half the maximum value, and also when it reaches the maximum value.
15	rh	0	The bit is set when the txmulticol_g counter reaches half the maximum value, and also when it reaches the maximum value.
14	rh	0	The bit is set when the txsinglecol_g counter reaches half the maximum value, and also when it reaches the maximum value.
13	rh	0	The bit is set when the txunderflowerror counter reaches half the maximum value, and also when it reaches the maximum value.
12	rh	0	The bit is set when the txbroadcastframes_gb counter reaches half the maximum value, and also when it reaches the maximum value.
11	rh	0	The bit is set when the txmulticastframes_gb counter reaches half the maximum value, and also when it reaches the maximum value.
10	rh	0	The bit is set when the txunicastframes_gb counter reaches half the maximum value, and also when it reaches the maximum value.

Table 30-13 MMC Transmit Interrupt Register (cont'd)

Bit	Access Type	Reset Value	Description
9	rh	0	The bit is set when the tx1024tomaxoctets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
8	rh	0	The bit is set when the tx512to1023octets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
7	rh	0	The bit is set when the tx256to511octets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
6	rh	0	The bit is set when the tx128to255octets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
5	rh	0	The bit is set when the tx65to127octets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
4	rh	0	The bit is set when the tx64octets_gb counter reaches half the maximum value, and also when it reaches the maximum value.
3	rh	0	The bit is set when the txmulticastframes_g counter reaches half the maximum value, and also when it reaches the maximum value.
2	rh	0	The bit is set when the txbroadcastframes_g counter reaches half the maximum value, and also when it reaches the maximum value.
1	rh	0	The bit is set when the txframecount_gb counter reaches half the maximum value, and also when it reaches the maximum value.
0	rh	0	The bit is set when the txoctetcount_gb counter reaches half the maximum value, and also when it reaches the maximum value.

MMC Receive Interrupt Mask Register

The MMC Receive Interrupt Mask register maintains the masks for the interrupts generated when receive statistic counters reach half their maximum value, and when they reach their maximum values. This register is 32 bits wide.

Table 30-14 MMC Receive Interrupt Mask Interrupt Register

Bit	Access Type	Reset Value	Description
31:24	r	00_H	Reserved
23	rw	0	Setting this bit masks the interrupt when the rxwatchdog counter reaches half the maximum value, and also when it reaches the maximum value.
22	rw	0	Setting this bit masks the interrupt when the rxvlanframes_gb counter reaches half the maximum value, and also when it reaches the maximum value.
21	rw	0	Setting this bit masks the interrupt when the rxfifooverflow counter reaches half the maximum value, and also when it reaches the maximum value.
20	rw	0	Setting this bit masks the interrupt when the rxpauseframes counter reaches half the maximum value, and also when it reaches the maximum value.
….	rw	…	… Same as above for corresponding counters in **MMC Receive Interrupt Register**
1	rw	0	Setting this bit masks the interrupt when the rxoctetcount_gb counter reaches half the maximum value, and also when it reaches the maximum value.
0	rw	0	Setting this bit masks the interrupt when the rxframecount_gb counter reaches half the maximum value, and also when it reaches the maximum value.

MMC Transmit Interrupt Mask Register

The MMC Transmit Interrupt Mask register maintains the masks for the interrupts generated when transmit statistic counters reach half their maximum value, and when they reach their maximum values. This register is 32 bits wide.

Table 30-15 MMC Transmit Interrupt Mask Register

Bit	Access Type	Reset Value	Description
31:25	r	00_H	Reserved
24	rw	0	Setting this bit masks the interrupt when the txvlanframes_g counter reaches half the maximum value, and also when it reaches the maximum value.

Table 30-15 MMC Transmit Interrupt Mask Register (cont'd)

Bit	Access Type	Reset Value	Description
23	rw	0	Setting this bit masks the interrupt when the txpauseframes counter reaches half the maximum value, and also when it reaches the maximum value.
22	rw	0	Setting this bit masks the interrupt when the txoexcessdef counter reaches half the maximum value, and also when it reaches the maximum value.
21	rw	0	Setting this bit masks the interrupt when the txframecount_g counter reaches half the maximum value, and also when it reaches the maximum value.
….	rw	…	… Same as above for corresponding counters in **MMC Transmit Interrupt Register**
1	rw	0	Setting this bit masks the interrupt when the txframecount_gb counter reaches half the maximum value, and also when it reaches the maximum value.
0	rw	0	Setting this bit masks the interrupt when the txoctetcount_gb counter reaches half the maximum value, and also when it reaches the maximum value.

MMC Receive Checksum Offload Interrupt Mask Register

The MMC Receive Checksum Offload Interrupt Mask register maintains the masks for the interrupts generated when the receive IPC (Checksum Offload) statistic counters reach half their maximum value , and when they reach their maximum values. This register is 32-bits wide.

Table 30-16 MMC Receive Checksum Offload Interrupt Mask Register

Bit	Access Type	Reset Value	Description
31:30	r	00	Reserved
29	rw	0	Setting this bit masks the interrupt when the rxicmp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.
28	rw	0	Setting this bit masks the interrupt when the rxicmp_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.

Table 30-16 MMC Receive Checksum Offload Interrupt Mask Register (cont'd)

Bit	Access Type	Reset Value	Description
27	rw	0	Setting this bit masks the interrupt when the rxtcp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.
26	rw	0	Setting this bit masks the interrupt when the rxtcp_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.
25	rw	0	Setting this bit masks the interrupt when the rxudp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.
24	rw	0	Setting this bit masks the interrupt when the rxudp_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.
23	rw	0	Setting this bit masks the interrupt when the rxipv6_nopay_octets counter reaches half the maximum value, and also when it reaches the maximum value.
22	rw	0	Setting this bit masks the interrupt when the rxipv6_hdrerr_octets counter reaches half the maximum value, and also when it reaches the maximum value.
21	rw	0	Setting this bit masks the interrupt when the rxipv6_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.
20	rw	0	Setting this bit masks the interrupt when the rxipv4_udsbl_octets counter reaches half the maximum value, and also when it reaches the maximum value.
19	rw	0	Setting this bit masks the interrupt when the rxipv4_frag_octets counter reaches half the maximum value, and also when it reaches the maximum value.
18	rw	0	Setting this bit masks the interrupt when the rxipv4_nopay_octets counter reaches half the maximum value, and also when it reaches the maximum value.
17	rw	0	Setting this bit masks the interrupt when the rxipv4_hdrerr_octets counter reaches half the maximum value, and also when it reaches the maximum value.

Table 30-16 MMC Receive Checksum Offload Interrupt Mask Register (cont'd)

Bit	Access Type	Reset Value	Description
16	rw	0	Setting this bit masks the interrupt when the rxipv4_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.
15:14	r	0	Reserved
13	rw	0	Setting this bit masks the interrupt when the rxicmp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.
12	rw	0	Setting this bit masks the interrupt when the rxicmp_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.
11	rw	0	Setting this bit masks the interrupt when the rxtcp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.
10	rw	0	Setting this bit masks the interrupt when the rxtcp_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.
9	rw	0	Setting this bit masks the interrupt when the rxudp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.
8	rw	0	Setting this bit masks the interrupt when the rxudp_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.
7	rw	0	Setting this bit masks the interrupt when the rxipv6_nopay_frms counter reaches half the maximum value, and also when it reaches the maximum value.
6	rw	0	Setting this bit masks the interrupt when the rxipv6_hdrerr_frms counter reaches half the maximum value, and also when it reaches the maximum value.
5	rw	0	Setting this bit masks the interrupt when the rxipv6_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.
4	rw	0	Setting this bit masks the interrupt when the rxipv4_udsbl_frms counter reaches half the maximum value, and also when it reaches the maximum value.

Table 30-16 MMC Receive Checksum Offload Interrupt Mask Register (cont'd)

Bit	Access Type	Reset Value	Description
3	rw	0	Setting this bit masks the interrupt when the rxipv4_frag_frms counter reaches half the maximum value, and also when it reaches the maximum value.
2	rw	0	Setting this bit masks the interrupt when the rxipv4_nopay_frms counter reaches half the maximum value, and also when it reaches the maximum value.
1	rw	0	Setting this bit masks the interrupt when the rxipv4_hdrerr_frms counter reaches half the maximum value, and also when it reaches the maximum value.
0	rw	0	Setting this bit masks the interrupt when the rxipv4_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.

MMC Receive Checksum Offload Interrupt Register

The MMC Receive Checksum Offload Interrupt register maintains the interrupts generated when receive IPC statistic counters reach half their maximum values (0x8000_0000), and when they cross their maximum values (0xFFFF_FFFF). When Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Receive Checksum Offload Interrupt register is 32 bits wide. When the MMC IPC counter that caused the interrupt is read, its corresponding interrupt bit is cleared. The counter's least-significant byte lane (bits[7:0]) must be read to clear the interrupt bit.

Note: Access restriction applies: Any read access to the register clears this register!

Table 30-17 MMC Receive Checksum Offload Interrupt Register

Bit	Access Type	Reset Value	Description
31:30	r	00	Reserved
29	rh	0	The bit is set when the rxicmp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.
28	rh	0	The bit is set when the rxicmp_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.

Table 30-17 MMC Receive Checksum Offload Interrupt Register (cont'd)

Bit	Access Type	Reset Value	Description
27	rh	0	The bit is set when the rxtcp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.
26	rh	0	The bit is set when the rxtcp_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.
25	rh	0	The bit is set when the rxudp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.
24	rh	0	The bit is set when the rxudp_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.
23	rh	0	The bit is set when the rxipv6_nopay_octets counter reaches half the maximum value, and also when it reaches the maximum value.
22	rh	0	The bit is set when the rxipv6_hdrerr_octets counter reaches half the maximum value, and also when it reaches the maximum value.
21	rh	0	The bit is set when the rxipv6_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.
20	rh	0	The bit is set when the rxipv4_udsbl_octets counter reaches half the maximum value, and also when it reaches the maximum value.
19	rh	0	The bit is set when the rxipv4_frag_octets counter reaches half the maximum value, and also when it reaches the maximum value.
18	rh	0	The bit is set when the rxipv4_nopay_octets counter reaches half the maximum value, and also when it reaches the maximum value.
17	rh	0	The bit is set when the rxipv4_hdrerr_octets counter reaches half the maximum value, and also when it reaches the maximum value.

Table 30-17 MMC Receive Checksum Offload Interrupt Register (cont'd)

Bit	Access Type	Reset Value	Description
16	rh	0	The bit is set when the rxipv4_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.
15:14	r	00	Reserved
13	rh	0	The bit is set when the rxicmp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.
12	rh	0	The bit is set when the rxicmp_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.
11	rh	0	The bit is set when the rxtcp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.
10	rh	0	The bit is set when the rxtcp_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.
9	rh	0	The bit is set when the rxudp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.
8	rh	0	The bit is set when the rxudp_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.
7	rh	0	The bit is set when the rxipv6_nopay_frms counter reaches half the maximum value, and also when it reaches the maximum value.
6	rh	0	The bit is set when the rxipv6_hdrerr_frms counter reaches half the maximum value, and also when it reaches the maximum value.
5	rh	0	The bit is set when the rxipv6_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.
4	rh	0	The bit is set when the rxipv4_udsbl_frms counter reaches half the maximum value, and also when it reaches the maximum value.

Table 30-17 MMC Receive Checksum Offload Interrupt Register (cont'd)

Bit	Access Type	Reset Value	Description
3	rh	0	The bit is set when the rxipv4_frag_frms counter reaches half the maximum value, and also when it reaches the maximum value.
2	rh	0	The bit is set when the rxipv4_nopay_frms counter reaches half the maximum value, and also when it reaches the maximum value.
1	rh	0	The bit is set when the rxipv4_hdrerr_frms counter reaches half the maximum value, and also when it reaches the maximum value.
0	rh	0	The bit is set when the rxipv4_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.

30.2.8 Power Management Block

This section describes the power management (PMT) mechanisms supported by the GMAC. PMT supports the reception of network (remote) wake-up frames and Magic Packet frames. PMT does not perform the clock gate function, but generates interrupts for wake-up frames and Magic Packets received by the GMAC. The PMT block sits on the receiver path of the GMAC and is enabled with remote wake-up frame enable and Magic Packet enable. These enables are in the PMT Control and Status register and are programmed by the Application. You can include the optional PMT module by selecting it in the coreKit during configuration. You have the option to select either or both types of power-management frames (remote wake-up and Magic Packet)

PMT registers are accessed in the same manner as with GMAC-CSR registers. Refer to registers 10 and 11, for mapping information.

When the power down mode is enabled in the PMT, then all received frames are dropped by the core and they are not forwarded to the application. The core comes out of the power down mode only when either a Magic Packet or a Remote Wake-up frame is received and the corresponding detection is enabled.

30.2.8.1 PMT Block Description

PMT Control and Status Register

The PMT CSR program the request wake-up events and monitor the wake-up events.

Note: Access restriction applies: Any read access to the register clears this register!

Table 30-18 PMT Control and Status Register

Bit	Access Type	Reset Value	Description
31	rwh	0	Wake-Up Frame Filter Register Pointer Reset When set, resets the Remote Wake-up Frame Filter register pointer to 3'b000. It is automatically cleared after 1 clock cycle. Access restriction applies: setting sets + clearing has no effect.
30:10	R	0000_00_H	Reserved
9	rw	00	Global Unicast When set, enables any unicast packet filtered by the GMAC (DAF) address recognition to be a wake-up frame.
8:7	R	0	Reserved
6	rh	0	Wake-Up Frame Received When set, this bit indicates the power management event was generated due to reception of a wake-up frame. This bit is cleared by a Read into this register.
5	rh	0	Magic Packet Received When set, this bit indicates the power management event was generated by the reception of a Magic Packet. This bit is cleared by a Read into this register.
4:3	R	00	Reserved
2	rw	0	Wake-Up Frame Enable When set, enables generation of a power management event due to wake-up frame reception.

Table 30-18 PMT Control and Status Register (cont'd)

Bit	Access Type	Reset Value	Description
1	rw	0	Magic Packet Enable When set, enables generation of a power management event due to Magic Packet reception.
0	rwh	0	Power Down When set, all received frames will be dropped. This bit is cleared automatically when a magic packet or Wake-Up frame is received, and Power-Down mode is disabled. Frames received after this bit is cleared are forwarded to the application.This bit must only be set when either the Magic Packet Enable, Global Unicast, or Wake-Up Frame Enable bit is set high. Access restriction applies: setting sets + clearing has no effect.

Remote Wake-Up Frame Filter Register

The register wkupfmfilter_reg, address (028_H), loads the Wake-up Frame Filter register. To load values in a Wake-up Frame Filter register, the entire register (wkupfmfilter_reg) must be written. The wkupfmfilter_reg register is loaded by sequentially loading the eight register values in address (028) for wkupfmfilter_reg0, wkupfmfilter_reg1,... wkupfmfilter_reg7, respectively. wkupfmfilter_reg is read in the same way.

Note: The internal counter to access the appropriate wkupfmfilter_reg is incremented when lane3 is accessed by the CPU. This should be kept in mind if you are accessing these registers in byte or half-word mode.

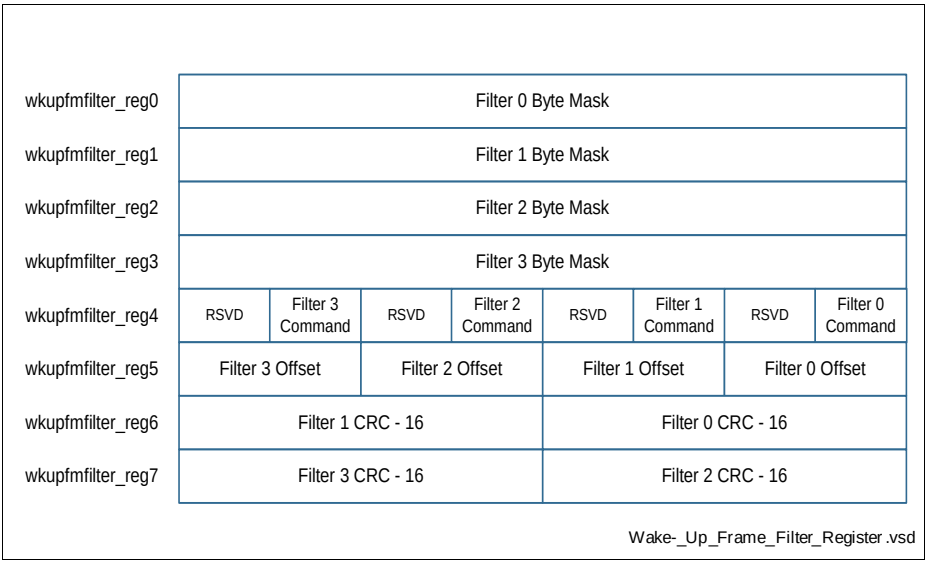


Figure 30-11 Wake-Up Frame Filter Register

Filter i Byte Mask

This register defines which bytes of the frame are examined by filter i (0, 1, 2, and 3) in order to determine whether or not the frame is a wake-up frame. The MSB (thirty-first bit) must be zero. Bit j [30:0] is the Byte Mask. If bit j (byte number) of the Byte Mask is set, then Filter i Offset + j of the incoming frame is processed by the CRC block; otherwise Filter i Offset + j is ignored.

Filter i Command

This 4-bit command controls the filter i operation. Bit 3 specifies the address type, defining the pattern's destination address type. When the bit is set, the pattern applies to only multicast frames; when the bit is reset, the pattern applies only to unicast frame. Bit 2 and Bit 1 are reserved. Bit 0 is the enable for filter i; if Bit 0 is not set, filter i is disabled.

Filter i Offset

This register defines the offset (within the frame) from which the frames are examined by filter i. This 8-bit pattern-offset is the offset for the filter i first byte to examined. The minimum allowed is 12, which refers to the 13th byte of the frame (offset value 0 refers to the first byte of the frame).

Filter i CRC-16

This register contains the CRC_16 value calculated from the pattern, as well as the byte mask programmed to the wake-up filter register block.

30.2.8.2 Remote Wake-Up Frame Detection

When the GMAC is in sleep mode and the remote wake-up bit is enabled in PMT Control and Status register (002C), normal operation is resumed after receiving a remote wake-up frame. The Application writes all eight wake-up filter registers, by performing a sequential Write to address (0028). The Application enables remote wake-up by writing a 1 to Bit 2 of the PMT Control and Status register.

PMT supports four programmable filters that allow support of different receive frame patterns. If the incoming frame passes the address filtering of Filter Command, and if Filter CRC-16 matches the incoming examined pattern, then the wake-up frame is received.

Filter_offset (minimum value 12, which refers to the 13th byte of the frame) determines the offset from which the frame is to be examined. Filter Byte Mask determines which bytes of the frame must be examined. The thirty-first bit of Byte Mask must be set to zero.

The remote wake-up CRC block determines the CRC value that is compared with Filter CRC-16. The wake-up frame is checked only for length error, FCS error, dribble bit error, GMII error, collision, and to ensure that it is not a runt frame. Even if the wake-up frame is more than 512 bytes long, if the frame has a valid CRC value, it is considered valid. Wake-up frame detection is updated in the PMT Control and Status register for every remote Wake-up frame received. A PMT interrupt to the Application triggers a Read to the PMT Control and Status register to determine reception of a wake-up frame.

30.2.8.3 Magic Packet Detection

The Magic Packet frame is based on a method that uses Advanced Micro Device's Magic Packet technology to power up the sleeping device on the network. The GMAC receives a specific packet of information, called a Magic Packet, addressed to the node on the network.

Only Magic Packets that are addressed to the device or a broadcast address will be checked to determine whether they meet the wake-up requirements. Magic Packets that pass the address filtering (unicast or broadcast) will be checked to determine whether they meet the remote Wake-on-LAN data format of 6 bytes of all ones followed by a GMAC Address appearing 16 times.

The application enables Magic Packet wake-up by writing a 1 to Bit 1 of the PMT Control and Status register. The PMT block constantly monitors each frame addressed to the node for a specific Magic Packet pattern. Each frame received is checked for a 48'hFF_FF_FF_FF_FF_FF pattern following the destination and source address field. The PMT block then checks the frame for 16 repetitions of the GMAC address without

any breaks or interruptions. In case of a break in the 16 repetitions of the address, the 48'hFF_FF_FF_FF_FF_FF pattern is scanned for again in the incoming frame. The 16 repetitions can be anywhere in the frame, but must be preceded by the synchronization stream (48'hFF_FF_FF_FF_FF_FF). The device will also accept a multicast frame, as long as the 16 duplications of the GMAC address are detected.

If the MAC address of a node is 48'h00_11_22_33_44_55, then the GMAC scans for the data sequence:

```
Destination Address Source Address …….. FF FF FF FF FF FF
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33
44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33
44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33
44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33
44 55
…CRC
```

Magic Packet detection is updated in the PMT Control and Status register for Magic Packet received. A PMT interrupt to the Application triggers a read to the PMT CSR to determine whether a Magic Packet frame has been received.

30.2.8.4 System Considerations During Power-Down

GMAC-UNIV neither gates nor stops clocks when Power-Down mode is enabled. Power saving by clock gating must be done outside the core by the application. The receive data path must be clocked with clk_rx_i during Power-Down mode, because it is involved in magic packet/wake-on-LAN frame detection. However, the transmit path and the application path clocks can be gated off during Power-Down mode.

The pmt_intr_o signal is asserted when a valid wake-up frame is received. This signal is generated in the clk_rx_i domain.

The recommended power-down and wake-up sequence is as follows.

1. Disable the Transmit DMA (if applicable) and wait for any previous frame transmissions to complete. These transmissions can be detected when Transmit Interrupt (TI–DMA Register 5[0]) is received.
2. Disable the MAC transmitter and MAC receiver by clearing the appropriate bits in the MAC Configuration register.
3. Wait until the Receive DMA empties all the frames from the Rx FIFO (a software timer may be required).
4. Enable Power-Down mode by appropriately configuring the PMT registers.
5. Enable the MAC Receiver and enter Power-Down mode.

6. Gate the application and transmit clock inputs to the core (and other relevant clocks in the system) to reduce power and enter Sleep mode.
7. On receiving a valid wake-up frame, the GMAC-UNIV asserts the pmt_intr_o signal and exits Power-Down mode.
8. On receiving the interrupt, the system must enable the application and transmit clock inputs to the core.
9. Read the PMT Status register to clear the interrupt, then enable the other modules in the system and resume normal operation.

30.2.9 Station Management Agent

The Station Management Agent (SMA) module allows the Application to access any PHY registers through a 2-wire Station Management interface (MIM). The interface supports accessing up to 32 PHYs.

The application can select one of the 32 PHYs and one of the 32 registers within any PHY and send control data or receive status information. Only one register in one PHY can be addressed at any given time. For more details on the communication from the Application to the PHYs, refer to the Reconciliation Sublayer and Media Independent Interface Specifications section of the IEEE 802.3z specification, 1000BASE Ethernet. The application sends the control data to the PHY and receives status information from the PHY through the SMA module, as shown in **Figure 30-12**.

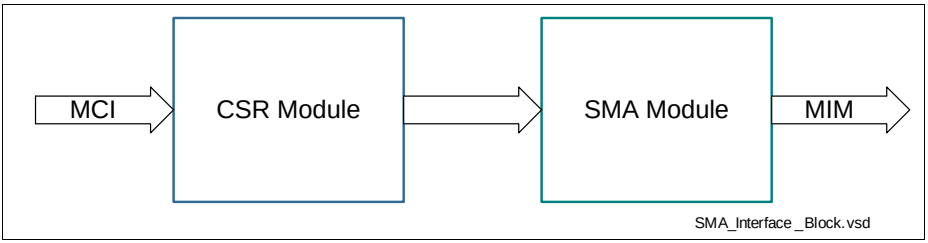


Figure 30-12 SMA Interface Block

30.2.9.1 Functions

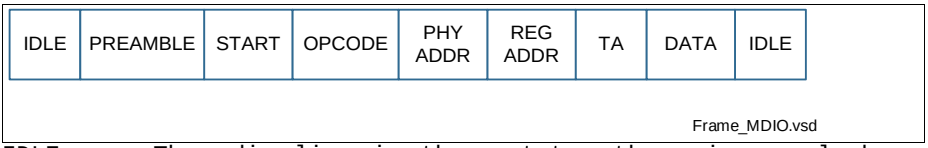
The GMAC initiates the Management Write/Read operation. The clock gmii_mdc_o is a divided clock from the Application clock clk_csr_i. The divide factor depends on the clock range setting in the GMII Address register. Clock range is set as follows:

Selection	clk_csr_i	MDC Clock
0000	60-100 MHz	clk_csr_i/42
0001	100-150 MHz	clk_csr_i/62
0010	20-35 MHz	clk_csr_i/16

Selection	clk_csr_i	MDC Clock
0011	35-60 MHz	clk_csr_i/26
0100	150-250 MHz	clk_csr_i/102
0101	250-300 MHz	clk_csr_i/124
0110, 0111	Reserved	

The gmii_mdc_o is the derivative of the Application clock clk_csr_i. The Management operation is performed through the gmii_mdi_i, gmii_mdo_o and gmii_mdo_o_e signals. A three-state buffer is implemented outside the GMAC to interface with an external PHY.

The frame structure on the MDIO line is shown below.




```
IDLE       The mdio line is three-state; there is no clock on
gmii_mdc_o
PREAMBLE   32 continuous bits of value 1
START      Start-of-frame is 2'01
OPCODE     2'b10 for Read and 2'b01 for Write
PHY ADDR   5-bit address select for one of 32 PHYs
REG ADDR   Register address in the selected PHY
TA         Turnaround is 2'bZ0 for Read and 2'b10 for Write
DATA       Any 16-bit value. In a Write operation, the GMAC drives
mdio; in a Read
           operation, PHY drives it.
```

30.2.9.2 MII Management Write Operation

When the user sets the GMII Write and Busy bits (see GMII Address Register, **"32-bit Register - GMII_Address" on Page 1-309**), the GMAC CSR module transfers the PHY address, the register address in PHY, and the write data (GMII Data Register) to the SMA to initiate a Write operation into the PHY registers. At this point, the SMA module starts a Write operation on the GMII Management Interface using the Management Frame Format specified in the GMII specifications (Section 22.2.4.5 of IEEE Standard). The application should not change the GMII Address register contents or the GMII Data register while the transaction is ongoing. Write operations to the GMII Address register or the GMII Data Register during this period are ignored (the Busy bit is high), and the transaction is completed without any error on the MCI interface.

After the Write operation has completed, the SMA indicates this to the CSR which then resets the Busy bit. The SMA module divides the CSR (Application) clock with the clock divider programmed (CR bits of GMII Address Register) to generate the MDC clock for this interface. The GMAC drives the MDIO line for the complete duration of the frame. The frame format for the Write operation is as follows:

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	1111...11	01	01	AAAAA	RRRRR	10	DDD . . . .DDD	Z

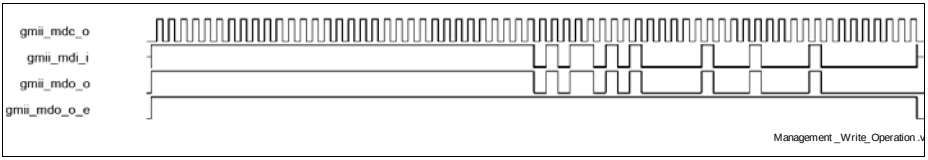


Figure 30-13 Management Write Operation

Figure 30-13 is a reference for the Write operation.

30.2.9.3 MII Management Read Operation

When the user sets the GMII Busy bit (see GMII Address Register, **“32-bit Register - GMII_Address” on Page 1-309**) with the GMII Write bit as 0, the GMAC CSR module transfers the PHY address and the register address in PHY to the SMA to initiate a Read operation in the PHY registers. At this point, the SMA module starts a Read operation on the GMII Management Interface using the Management Frame Format specified in the GMII specifications (Section 22.2.4.5 of IEEE Standard). The application should not change the GMII Address register contents or the GMII Data register while the transaction is ongoing. Write operations to the GMII Address register or GMII Data Register during this period are ignored (the Busy bit is high) and the transaction completed without any error on the MCI interface.

After the Read operation has completed, the SMA indicates this to the CSR, which then resets the Busy bit and updates the GMII Data register with the data read from the PHY. The SMA module divides the CSR (Application) clock with the clock divider programmed (CR bits of GMII Address Register) to generate the MDC clock for this interface. The GMAC drives the MDIO line for the complete duration of the frame except during the Data fields when the PHY is driving the MDIO line. The frame format for the Read operation is as follows:

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	1111...11	01	10	AAAAA	RRRRR	Z0	DDD . . . .DDD	Z

Figure 30-14 is a reference for the read operation.

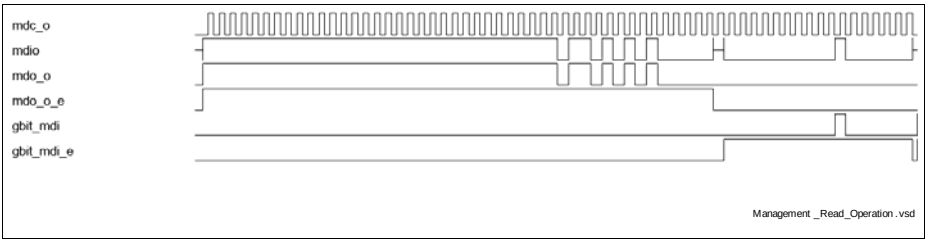


Figure 30-14 Management Read Operation

30.2.10 Reduced Media Independent Interface

The Reduced Media Independent Interface (RMII) specification reduces the pin count between Ethernet PHYs and Switch ASICs (only in 10/100 mode). According to the IEEE 802.3u standard, an MII contains 16 pins for data and control. In devices incorporating multiple MAC or PHY interfaces (such as switches), the number of pins adds significant cost with increase in port count. The RMII specification addresses this problem by reducing the pin count to 7 for each port — a 62.5% decrease in pin count.

- The RMII module is instantiated between the GMAC and the PHY. This helps translation of the MAC's MII into the RMII. The RMII block has the following characteristics:
- Supports 10 Mbit/s and 100 Mbit/s operating rates.
- Two clock references are sourced externally, providing independent, 2-bit wide transmit and receive paths.

30.2.10.1 Block Diagram

Figure 30-15 shows the position of the RMII block relative to the Ethernet MAC and RMII PHY. The RMII block is placed in front of the Ethernet MAC to translate the MII signals to RMII signals.

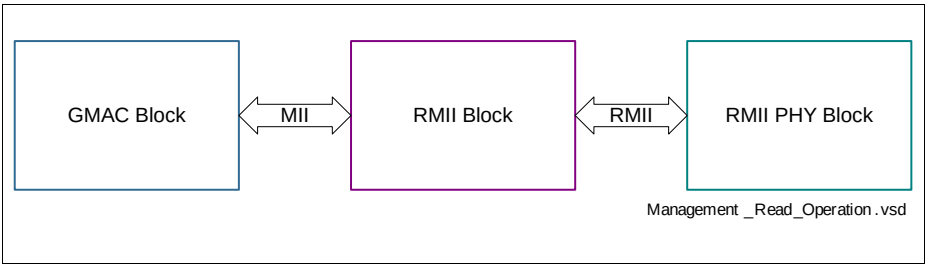


Figure 30-15 RMII Block Diagram

30.2.10.2 Block Overview

The following list describes the RMII's hardware components, which are shown in **Figure 30-16**. Each of these blocks is briefly described in the following sections.

MII-RMII Transmit (MRT) Block: This block translates all MII transmit signals to RMII transmit signals. All RMII signals are synchronous to clk_rmii_i.

MII-RMII Receive (MRR) Block: This block translates all RMII receive signals to MII receive signals. All MII signals are synchronous to clk_rx_i. You must ensure that the same clock is connected to both clk_tx_i and clk_rx_i clock input ports in RMII mode. This is required because clock-MUXing logic is avoided inside the GMAC core.

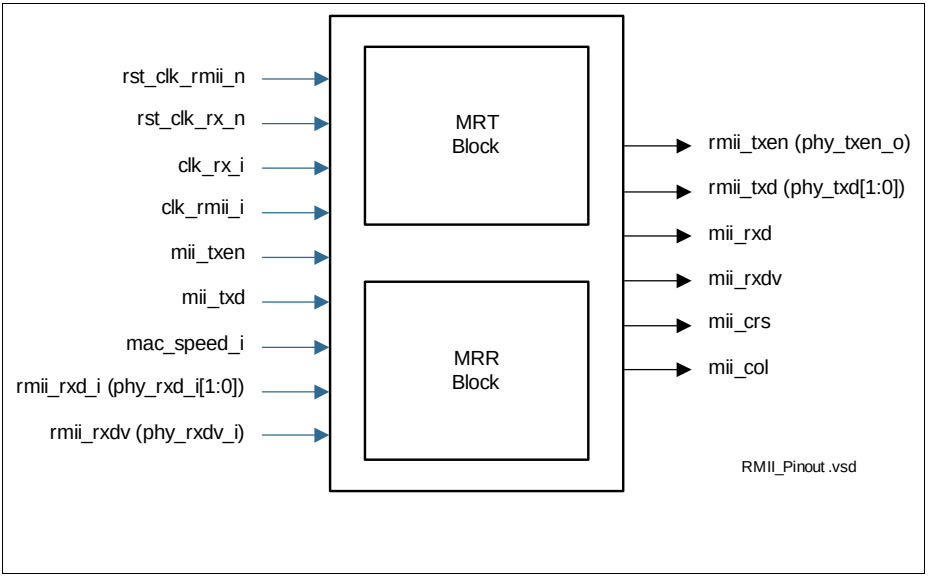


Figure 30-16 RMII Pinout

Note: The mac_speed_i signal configures the RMII to operate at 10 Mbit/s or 100 Mbit/s. This signal is driven from the MAC Configuration register's FES bit.

30.2.10.3 Transmit Bit Ordering

Each nibble from the MII must be transmitted on the RMII a di-bit at a time with the order of di-bit transmission shown in **Figure 30-17**. The lower order bits (D1 and D0) are transmitted first followed by higher order bits (D2 and D3).

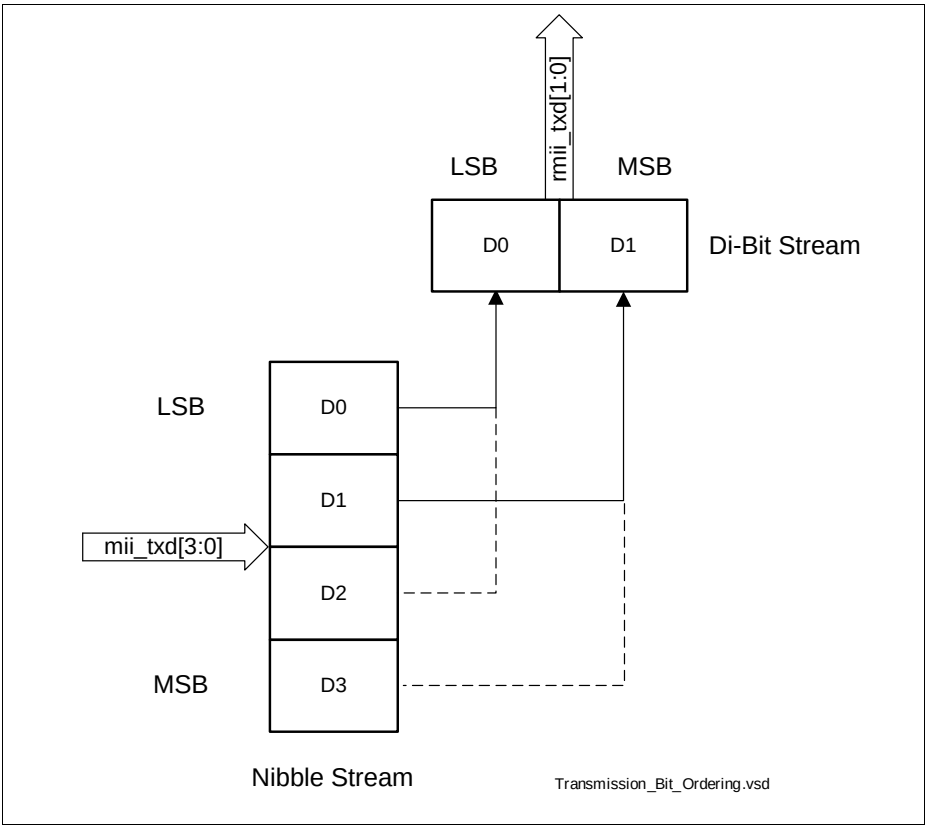


Figure 30-17 Transmission Bit Ordering

30.2.10.4 RMII Transmit Timing Diagrams

Figure 30-18 through **Figure 30-21** show MII-to-RMII transaction timing.

Figure 30-18 shows the start of MII transmission and the following RMII transmission in 100 Mbit/s mode.

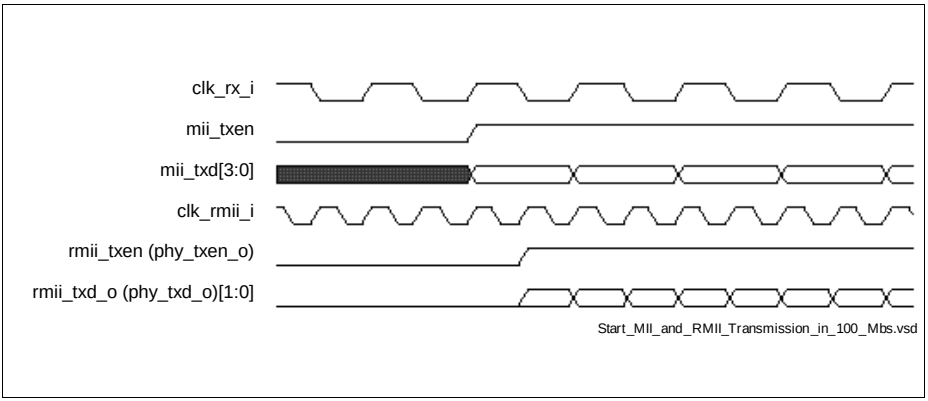


Figure 30-18 Start of MII and RMII Transmission in 100 Mbit/s Mode

Figure 30-19 shows the end of frame transmission for MII and RMII in 100 Mbit/s mode.

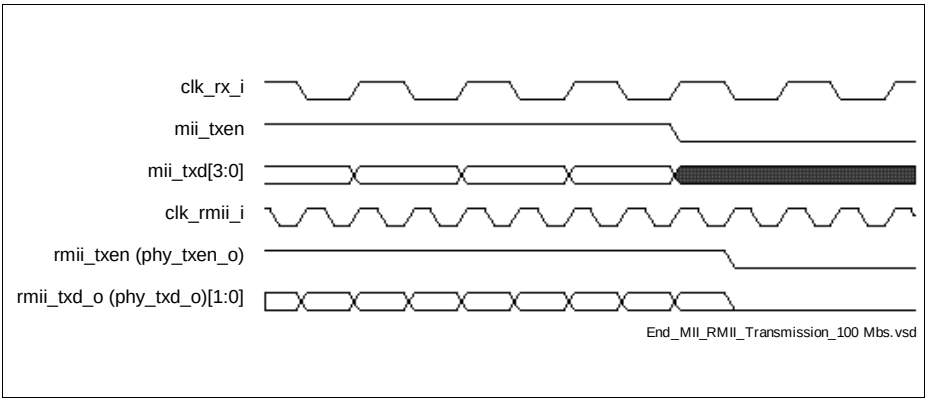


Figure 30-19 End of MII and RMII Transmission in 100 Mbit/s Mode

Figure 30-20 shows the start of MII transmission and the following RMII transmission in 10 Mbit/s mode.

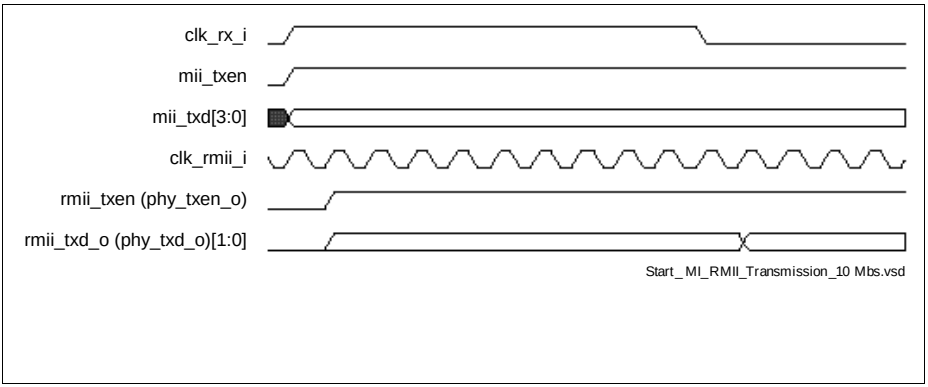


Figure 30-20 Start of MII and RMII Transmission in 10 Mbit/s Mode

Figure 30-21 shows the end of MII transmission and RMII transmission in 10 Mbit/s mode.

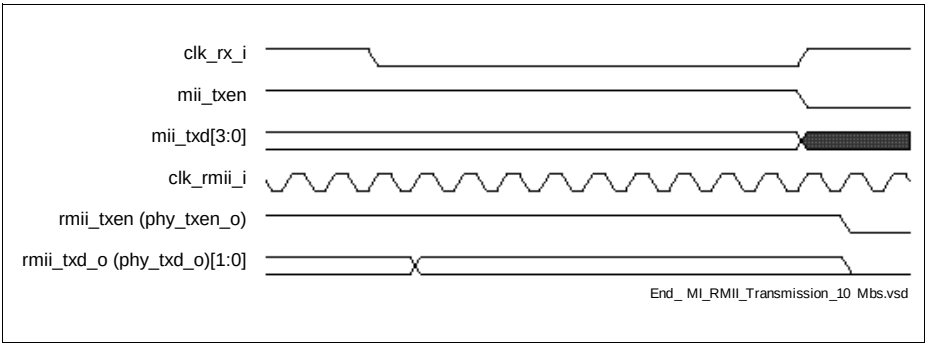


Figure 30-21 End of MII and RMII Transmission in 10 Mbit/s Mode

Receive Bit Ordering

Each nibble is transmitted to the MII from the di-bit received from the RMII in the nibble transmission order shown in **Figure 30-22**. The lower order bits (D0 and D1) are received first, followed by the higher order bits (D2 and D3).

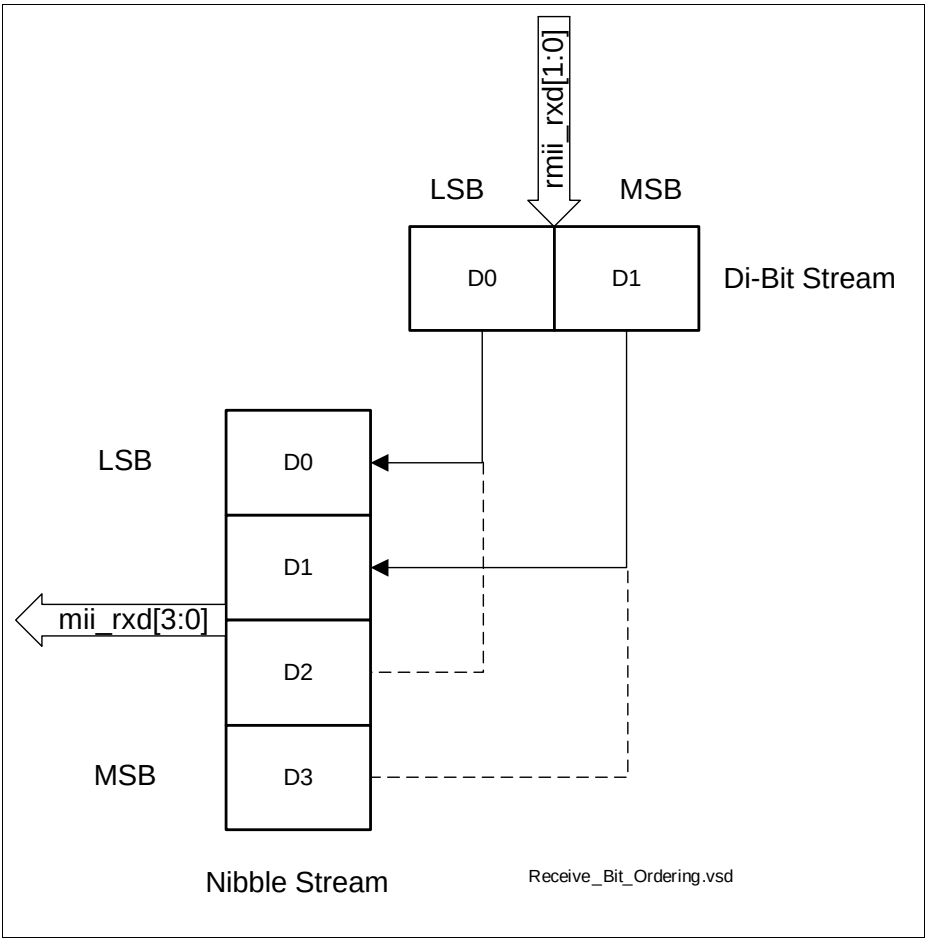


Figure 30-22 Receive Bit Ordering

30.2.11 Interrupts From the GMAC Core

Interrupts can be generated from the GMAC core as a result of various events in the optional modules in it.

The Interrupt Status register in the GMAC Register Map (**"GMAC Register Map" on Page 1-281**) describes the events that can cause an interrupt from the GMAC core. Each event can be prevented from asserting the interrupt by setting the corresponding mask bits in the Interrupt Mask register.

The interrupt register bits only indicate the block from which the event is reported. You must read the corresponding status registers and other registers to clear the interrupt. For example, Bit 0 of the Interrupt register, set high, indicates that the link status on the RGMII interface has changed. You must read Register 54 (the RGMII Status register) to clear this interrupt event.

The interrupts from the RGMII and PCS blocks are combined (OR'ed) and given as the GLI bit in the DMA Status register. The TTI, GPI, and GMI signals in **Figure 30-23** refer to the bits that can be read in the DMA Status register.

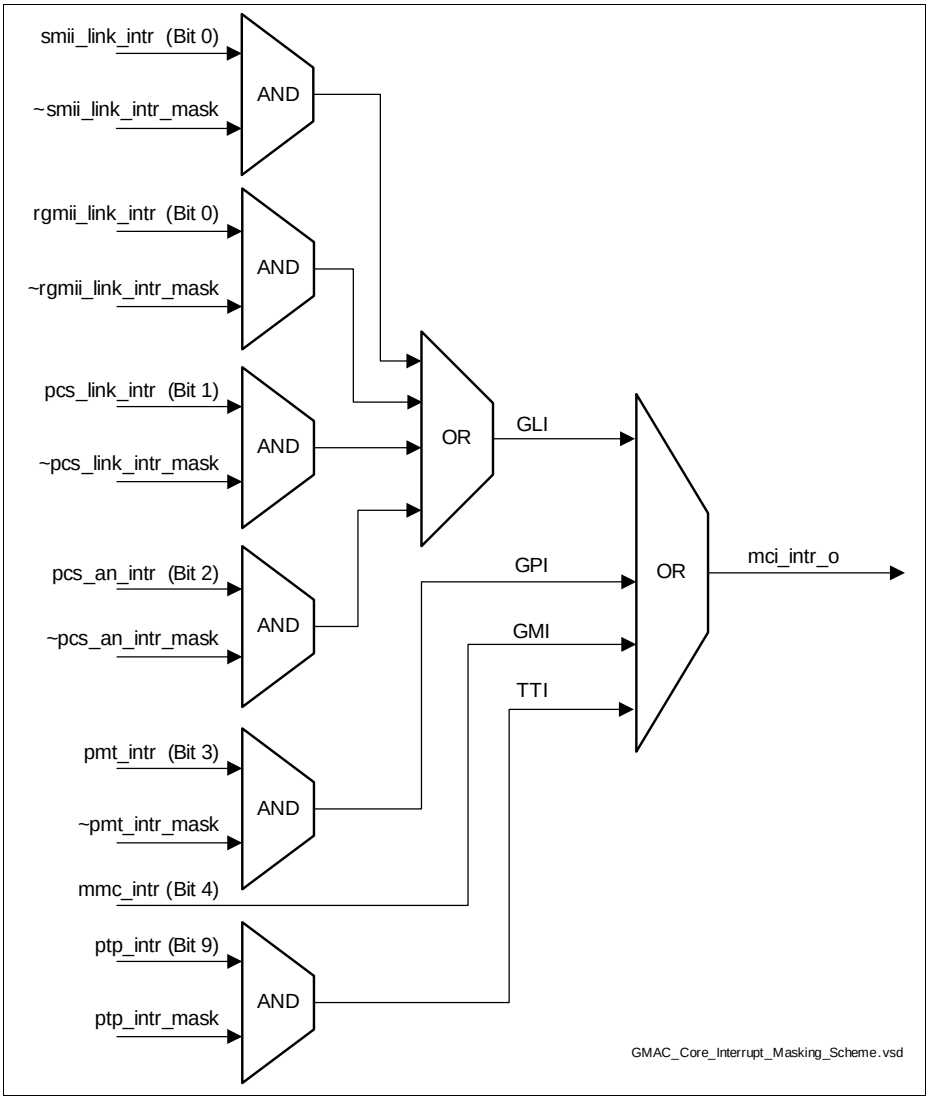


Figure 30-23 GMAC Core Interrupt Masking Scheme

30.3 Register

General notes on registers:

- All CSRs are implemented as 32-bit registers and can be accessed in 1 read/write cycle with a 32-bit MCI interface (or 32-bit AHB slave port). If you initiate a non-32-bit register access, the byte enable signals (mci_be_i[3:0]) qualify the corresponding register bytes. Only write operations are qualified with byte enables, while read operations are always 32-bit, unless the register is affected by a read operation. Byte enable signals qualify reads to such registers as, for example, the MMC counter registers.
- The transfer of particular register contents (e.g., MAC DA address register) to the Transmit or Receive clock domains are initiated when a particular byte is written. This is indicated in the register descriptions, where applicable.
- When any Register content is being transferred to a different clock domain after a write operation, there should not be any further writes to the same location until the first write is updated. Otherwise, the second write operation will not get updated to the destination clock domain. Thus the delay between two writes to the same register location should be at least 4 cycles of the destination clock (PHY receive clock or PHY transmit clock).

30.3.1 Register Maps

The register maps in this section provide high-level summaries of each register or group of registers. The detailed register descriptions in **“Register Description” on Page 1-292**.

Reserved address space (marked as “do not use”) will generate no bus error (nBE).

All registers in the DMA and MAC Registers map (1000_H - 2058_H) are P = ACCEN protected. The registers for additional Module Control are ACCEN protected according to **Figure 30-22**).

30.3.1.1 Register Overview

Table 30-19 Registers Address Space - ETH Module

Module	Base Address	End Address	Note
ETH	F001 $D000_H$	F001 $F0FF_H$	-

30.3.1.2 DMA Register Map

Table 30-20 DMA Register Map

DMA Register No.	Offset Address	Register Name and Description
0	2000_H	**DMA Register 0 (Bus Mode Register)** Controls the Host Interface Mode.
1	2004_H	**DMA Register 1 (Transmit Poll Demand Register)** Used by the host to instruct the DMA to poll the Transmit Descriptor List.
2	2008_H	**DMA Register 2 (Receive Poll Demand Register)** Used by the Host to instruct the DMA to poll the Receive Descriptor List.
3	$200C_H$	**DMA Register 3 (Receive Descriptor List Address Register)** Points the DMA to the start of the Receive Descriptor list.
4	2010_H	**DMA Register 4 (Transmit Descriptor List Address Register)** Points the DMA to the start of the Transmit Descriptor List.
5	2014_H	**DMA Register 5 (Status Register)** The Software driver (application) reads this register during interrupt service routine or polling to determine the status of the DMA.
6	2018_H	**DMA Register 6 (Operation Mode Register)** Establishes the Receive and Transmit operating modes and command.
7	$201C_H$	**DMA Register 7 (Interrupt Enable Register)** Enables the interrupts reported by the Status Register.
8	2020_H	**DMA Register 8 (Missed Frame and Buffer Overflow Counter Register)** Contains the counters for discarded frames because no host Receive Descriptor was available, and discarded frames because of Receive FIFO Overflow.
9	2024_H	**DMA Register 9 (Receive Interrupt Watchdog Timer Register)** Watchdog time-out for Receive Interrupt (RI) from DMA
12-17	2030_H-2044_H	Do not use

Table 30-20 DMA Register Map

DMA Register No.	Offset Address	Register Name and Description
18	2048_H	**DMA Register 18 (Current Host Transmit Descriptor Register)** Points to the start of current Transmit Descriptor read by the DMA.
19	$204C_H$	**DMA Register 19 (Current Host Receive Descriptor Register)** Points to the start of current Receive Descriptor read by the DMA.
20	2050_H	**DMA Register 20 (Current Host Transmit Buffer Address Register)** Points to the current Transmit Buffer address read by the DMA.
21	2054_H	**DMA Register 21 (Current Host Receive Buffer Address Register)** Points to the current Receive Buffer address read by the DMA.
22	2058_H	**DMA Register 22 (HW Feature Register)** Indicates the presence of the optional features of the core.
23-63	$205C_H$-$20FC_H$	Do not use

30.3.1.3 GMAC Register Map

Table 1-59 provides the address map of the GMAC core registers.

Table 30-21 GMAC Register Map

GMAC Register No.	Offset Address	Register Name and Description
0	1000_H	**GMAC Register 0 (MAC Configuration Register)** This is the operation mode register for the MAC.
1	1004_H	**GMAC Register 1 (MAC Frame Filter)** Contains the frame filtering controls.

Table 30-21 GMAC Register Map

GMAC Register No.	Offset Address	Register Name and Description
2	1008_H	**GMAC Register 2 (Hash Table High Register)** Contains the higher 32 bits of the Multicast Hash table.This register is present only when the Hash filter function is selected in coreConsultant. (See Table ??)
3	$100C_H$	**GMAC Register 3 (Hash Table Low Register)** Contains the lower 32 bits of the Multicast Hash table. This register is present only when the Hash filter function is selected in coreConsultant. (See Table ??)
4	1010_H	**GMAC Register 4 (GMII Address Register)** Controls the management cycles to an external PHY. This register is present only when the Station Management (MDIO) feature is selected in coreConsultant. (See Table ??)
5	1014_H	**GMAC Register 5 (GMII Data Register)** Contains the data to be written to or read from the PHY register. This register is present only when the Station Management (MDIO) feature is selected in coreConsultant. (See Table ??)
6	1018_H	**GMAC Register 6 (Flow Control Register)** Controls the generation of control frames.
7	$101C_H$	**GMAC Register 7 (VLAN Tag Register)** Identifies IEEE 802.1Q VLAN type frames.
8	1020_H	**GMAC Register 8 (Version Register)** Identifies the version of the Core
9	1024_H	**GMAC Register 9 (Debug Register)** Gives the status of various internal blocks for debugging

Table 30-21 GMAC Register Map

GMAC Register No.	Offset Address	Register Name and Description
10	1028_H	**Remote Wake-Up Frame Filter** This is the address through which the remote Wake-up Frame Filter registers (wkupfmfilter_reg) are written/read by the Application. wkupfmfilter_reg is actually a pointer to eight (not transparent) such wkupfmfilter_reg registers. Eight sequential Writes to this address (028) will write all wkupfmfilter_reg registers. Eight sequential Reads from this address (028) will read all wkupfmfilter_reg registers. This register contains the higher 16 bits of the 7th MAC address. This register is present only when the PMT module Remote Wake-up feature is selected in coreConsultant. (See Table ??) Refer to “Remote Wake-Up Frame Filter Register” on page ?? for additional information.
11	$102C_H$	**PMT Control and Status** This register is present only when the PMT module is selected in coreConsultant. (See Table ??) Refer to “PMT Control and Status Register” on page ?? for additional information.
12-13	1030_H - 1034_H	Do not use
14	1038_H	**GMAC Register 14 (Interrupt Status Register)** Contains the interrupt status.
15	$103C_H$	**GMAC Register 15 (Interrupt Mask Register)** Contains the masks for generating the interrupts.
16	1040_H	**GMAC Register 16 (MAC Address 0 High Register)** Contains the higher 16 bits of the first MAC address.
17	1044_H	**GMAC Register 17 (MAC Address 0 Low Register)** Contains the lower 32 bits of the first MAC address.
18	1048_H	**GMAC Register 18 (MAC Address 1 High Register)** Contains the higher 16 bits of the second MAC address. This register is present only when Enable MAC Address 1 is selected in coreConsultant. (See Table ??).

Table 30-21 GMAC Register Map

GMAC Register No.	Offset Address	Register Name and Description
19	$104C_H$	**GMAC Register 19 (MAC Address 1 Low Register)** Contains the lower 32 bits of the second MAC address. This register is present only when Enable MAC Address 1 is selected in coreConsultant. (See Table ??).
20	1050_H	**MAC Address 2 High Register** Contains the higher 32 bits of the third MAC address. This register is present only when Enable MAC Address 2 is selected in coreConsultant. (See Table ??).
21	1054_H	**MAC Address 2 Low Register** Contains the lower 32 bits of the third MAC address. This register is present only when Enable MAC Address 2 is selected in coreConsultant. (See Table ??).
22	1058_H	**MAC Address 3 High Register** Contains the higher 16 bits of the fourth MAC address. This register is present only when Enable MAC Address 3 is selected in coreConsultant. (See Table ??).
23	$105C_H$	**MAC Address 3 Low Register** Contains the lower 32 bits of the fourth MAC address. This register is present only when Enable MAC Address 3 is selected in coreConsultant. (See Table ??).
55-63	$10DC_H$ - $10FC_H$	Do not use
64-191	1100_H - $12FC_H$	MMC Register Map See **Table 1-17 “MMC Register Map” on Page 1-131**.
192-447	1300_H - $16FC_H$	Do not use
448	1700_H	**Register 448 (Time Stamp Control Register)** Controls the time stamp generation and update logic.
449	1704_H	**Register 449 (Sub-Second Increment Register)** Contains the 8-bit value by which the Sub-Second register is incremented.This register is only present when IEEE1588 time stamping is enabled without an external time stamp input.

Table 30-21 GMAC Register Map

GMAC Register No.	Offset Address	Register Name and Description
450	1708_H	**Register 450 (System Time - Seconds Register)** Contains the lower 32 bits of the seconds field of the system time. This register is only present when IEEE1588 time stamping is enabled without an external time stamp input.
451	$170C_H$	**Register 451 (System Time - Nanoseconds Register)** Contains 32 bits of the nanoseconds field of the system time. This register is only present when IEEE1588 time stamping is enabled without an external time stamp input.
452	1710_H	**Register 452 (System Time - Seconds Update Register)** Contains the lower 32 bits of the seconds field to be written to, added to, or subtracted from the System Time value. This register is only present when IEEE1588 time stamping is enabled without an external time stamp input.
453	1714_H	**Register 453 (System Time - Nanoseconds Update Register)** Contains 32 bits of the nanoseconds field to be written to, added to, or subtracted from the System Time value. This register is only present when IEEE1588 time stamping is enabled without an external time stamp input.
454	1718_H	**Register 454 (Time Stamp Addend Register)** This register is used by the software to readjust the clock frequency linearly to match the master clock frequency. This register is only present when IEEE1588 time stamping is enabled without an external time stamp input.
455	$171C_H$	**Register 455 (Target Time Seconds Register)** Contains the higher 32 bits of time to be compared with the system time for interrupt event generation. This register is only present when IEEE1588 time stamping is enabled without an external time stamp input.
456	1720_H	**Register 456 (Target Time Nanoseconds Register)** Contains the lower 32 bits of time to be compared with the system time for interrupt event generation. This register is only present when IEEE1588 time stamping is enabled without an external time stamp input.

Table 30-21 GMAC Register Map

GMAC Register No.	Offset Address	Register Name and Description
457	1724_H	**Register 457 (System Time - Higher Word Seconds Register)** Contains the most significant 16-bits of the time stamp seconds value. This register is optional and can be selected using the coreKit parameter identified in “IEEE 1588 Time Stamp Block” on page ??.
458	1728_H	**Register 458 (Time Stamp Status Register)** Contains the PTP status. This register is available only when the advanced IEEE 1588 timestamp feature is selected.
459	$172C_H$	**Register 459 (PPS Control Register)** This register is used to control the interval of the PPS signal output, such that a duration of less than 1 second can be achieved between pulses.
460	1730_H	**Register 460 (Auxiliary Time Stamp - Nanoseconds Register)** Contains the lower 32 bits (nanoseconds field) of the auxiliary time stamp register.
461	1734_H	**Register 461 (Auxiliary Time Stamp - Seconds Register)** Contains the upper 32 bits (seconds field) of the auxiliary time stamp register.
462-511	1738_H - $17FC_H$	Do not use
512-511	1738_H - $17FC_H$	Do not use

30.3.1.4 Ethernet MAC Additional Module Control Registers

This section describes the additional module control registers of the Ethernet MAC module. All Ethernet MAC register names described in this section will be referenced in other parts of the TC21x/TC22x/TC23x User's Manual by the module name prefix "Ethernet MAC_" for the Ethernet MAC interface.

All registers in the Ethernet MAC address spaces are reset with the application reset (definition see SCU section "Reset Operation"). P = ACCEN protection

Table 30-22 Register Overview - ETH Add. Mod. Contr. Registers

Register Short Name	Register Long Name	Offset [1]	Access Mode		Description see
			Read	Write	
Module Control Registers					
ETH_CLC	Clock Control Register	0000_H	SV, U	SV, E, P	Page 30-462
ETH_ID	Module Identification Register	0004_H	SV, U	nBE	Page 30-461
ETH_GPCTL	General Purpose Control Register	0008_H	SV, U	SV	Page 30-463
BPI Kernel Registers					
ETH_ACCEN0	Access Enable Register 0	$000C_H$	U, SV	SV, SE	Page 30-465
ETH_ACCEN1	Access Enable Register 1	0010_H	U, SV	SV, SE	Page 30-466
ETH_KRST0	Reset Control Register 0	0014_H	U, SV	SV, P	Page 30-467
ETH_KRST1	Reset Control Register 1	0018_H	U, SV	SV, P	Page 30-468
ETH_KRSTCLR	Reset Status Clear Reg.	$001C_H$	U, SV	SV, P	Page 30-469
-	reserved	0020_H - $0FFF_H$	nBE	nBE	

1) The absolute register address is calculated as follows:
Module Base Address (**Table 1-60**) + Offset Address (shown in this column)

30.3.2 Register Description

This section defines the bits for each register.

32-bit Register - MAC_Configuration

The MAC Configuration register establishes receive and transmit operating modes.

ETH_MAC_CONFIGURATION
Register 0 - MAC Configuration Register (1000_H) **Reset Value: $0000\ 8000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED_31	SARC			TWOKPE	SFTERR	CST	TC	WD	JD	BE	JE	IFG			DCRS
r	r			rw	r	rw	r	rw	rw	r	rw	rw			rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS	FES	DO	LM	DM	IPC	DR	LUD	ACS	BL		DC	TE	RE	PRELEN	
r	rw	rw	rw	rw	rw	rw	r	rw	rw		rw	rw	rw	rw	

Field	Bits	Type	Description
PRELEN	[1:0]	rw	**Preamble Length for Transmit Frames** These bits control the number of preamble bytes that are added to the beginning of every Transmit frame. The preamble reduction occurs only when the MAC is operating in the full-duplex mode. * 2'b00: 7 bytes of preamble * 2'b01: 5 byte of preamble * 2'b10: 3 bytes of preamble * 2'b11: reserved
RE	2	rw	**Receiver Enable** When this bit is set, the receiver state machine of the MAC is enabled for receiving frames from the GMII or MII. When this bit is reset, the MAC receive state machine is disabled after the completion of the reception of the current frame, and does not receive any further frames from the GMII or MII.

Field	Bits	Type	Description
TE	3	rw	**Transmitter Enable** When this bit is set, the transmit state machine of the MAC is enabled for transmission on the GMII or MII. When this bit is reset, the MAC transmit state machine is disabled after the completion of the transmission of the current frame, and does not transmit any further frames.
DC	4	rw	**Deferral Check** When this bit is set, the deferral check function is enabled in the MAC. The MAC issues a Frame Abort status, along with the excessive deferral error bit set in the transmit frame status, when the transmit state machine is deferred for more than 24,288 bit times in the 10 or 100 Mbps mode. If the MAC is configured for 1000 Mbps operation, or if the Jumbo frame mode is enabled in the 10 or 100 Mbps mode, the threshold for deferral is 155,680 bits times. Deferral begins when the transmitter is ready to transmit, but is prevented because of an active carrier sense signal (CRS) on GMII or MII. Defer time is not cumulative. When the transmitter defers for 10,000 bit times, it transmits, collides, backs off, and then defers again after completion of back-off. The deferral timer resets to 0 and restarts. When this bit is reset, the deferral check function is disabled and the MAC defers until the CRS signal goes inactive. This bit is applicable only in the half-duplex mode and is reserved (RO) in the full-duplex-only configuration.

Field	Bits	Type	Description
BL	[6:5]	rw	**Back-Off Limit** The Back-Off limit determines the random integer number (r) of slot time delays (4,096 bit times for 1000 Mbps and 512 bit times for 10/100 Mbps) for which the MAC waits before rescheduling a transmission attempt during retries after a collision. This bit is applicable only in the half-duplex mode and is reserved (RO) in the full-duplex-only configuration. * 00: k = min (n, 10) * 01: k = min (n, 8) * 10: k = min (n, 4) * 11: k = min (n, 1) where n = retransmission attempt. The random integer r takes the value in the range 0 <= r < kth power of 2
ACS	7	rw	**Automatic Pad or CRC Stripping** When this bit is set, the MAC strips the Pad or FCS field on the incoming frames only if the value of the length field is less than 1,536 bytes. All received frames with length field greater than or equal to 1,536 bytes are passed to the application without stripping the Pad or FCS field. When this bit is reset, the MAC passes all incoming frames, without modifying them, to the Host.
LUD	8	r	**Link Up or Down** This bit indicates whether the link is up or down during the transmission of configuration in the RGMII, SGMII, or SMII interface: * 0: Link Down * 1: Link Up This bit is reserved (RO with default value) and is enabled when the RGMII, SGMII, or SMII interface is enabled during core configuration.

Field	Bits	Type	Description
DR	9	rw	**Disable Retry** When this bit is set, the MAC attempts only one transmission. When a collision occurs on the GMII or MII interface, the MAC ignores the current frame transmission and reports a Frame Abort with excessive collision error in the transmit frame status. When this bit is reset, the MAC attempts retries based on the settings of the BL field (Bits [6:5]). This bit is applicable only in the half-duplex mode and is reserved (RO with default value) in the full-duplex-only configuration.
IPC	10	rw	**Checksum Offload** When this bit is set, the MAC calculates the 16-bit one's complement of the one's complement sum of all received Ethernet frame payloads. It also checks whether the IPv4 Header checksum (assumed to be bytes 2526 or 2930 (VLAN-tagged) of the received Ethernet frame) is correct for the received frame and gives the status in the receive status word. The MAC also appends the 16-bit checksum calculated for the IP header datagram payload (bytes after the IPv4 header) and appends it to the Ethernet frame transferred to the application (when Type 2 COE is deselected). When this bit is reset, this function is disabled. When Type 2 COE is selected, this bit, when set, enables the IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking. When this bit is reset, the COE function in the receiver is disabled and the corresponding PCE and IP HCE status bits are always cleared. If the IP Checksum Offload feature is not enabled during core configuration, this bit is reserved (RO with default value).
DM	11	rw	**Duplex Mode** When this bit is set, the MAC operates in the full-duplex mode where it can transmit and receive simultaneously. This bit is RO with default value of 1'b1 in the full-duplex-only configuration.

Field	Bits	Type	Description
LM	12	rw	**Loopback Mode** When this bit is set, the MAC operates in the loopback mode at GMII or MII. The (G)MII Receive clock input (clk_rx_i) is required for the loopback to work properly, because the Transmit clock is not looped-back internally.
DO	13	rw	**Disable Receive Own** When this bit is set, the MAC disables the reception of frames when the gmii_txen_o is asserted in the half-duplex mode. When this bit is reset, the MAC receives all packets that are given by the PHY while transmitting. This bit is not applicable if the MAC is operating in the full-duplex mode. This bit is reserved (RO with default value) if the MAC is configured for the full-duplex-only operation.
FES	14	rw	**Speed** This bit selects the speed in the MII, RMII, SMII, RGMII, SGMII, or RevMII interface: * 0: 10 Mbps * 1: 100 Mbps This bit is reserved (RO) by default and is enabled only when the RMII, SMII, RGMII, SGMII, or RevMII interface is enabled during core configuration. This bit generates link speed encoding when TC (Bit 24) is set in the RGMII, SMII, or SGMII mode. This bit is always driven for RevMII. If the MAC is configured with an RMII, SMII, SGMII, or RGMII PHY interface, this bit can optionally be driven as an output signal (mac_speed_o[0]) to reflect the value of this bit in the mac_speed_o signal. In addition, this bit is reserved when RMII is enabled without enabling the Output Port for speed selection option during core configuration.

Field	Bits	Type	Description
PS	15	r	**Port Select** This bit selects between GMII and MII: * 0: GMII (1000 Mbps) * 1: MII (10/100 Mbps) In the 10 or 100 Mbps-only (always 1) or 1000 Mbps-only (always 0) configurations, this bit is read-only with the appropriate value. In the default 10/100/1000 Mbps configurations, this bit is R_W. The mac_portselect_o signal reflects the value of this bit.
DCRS	16	rw	**Disable Carrier Sense During Transmission** When set high, this bit makes the MAC transmitter ignore the (G)MII CRS signal during frame transmission in the half-duplex mode. This request results in no errors generated because of Loss of Carrier or No Carrier during such transmission. When this bit is low, the MAC transmitter generates such errors because of Carrier Sense and can even abort the transmissions. This bit is reserved (and RO) in the full-duplex-only configurations.
IFG	[19:17]	rw	**Inter-Frame Gap** These bits control the minimum IFG between frames during transmission. * 000: 96 bit times * 001: 88 bit times * 010: 80 bit times * ... * 111: 40 bit times In the half-duplex mode, the minimum IFG can be configured only for 64 bit times (IFG = 100). Lower values are not considered. In the 1000-Mbps mode, the minimum IFG supported is 64 bit times (and above) in the GMAC-CORE configuration and 80 bit times (and above) in other configurations.
JE	20	rw	**Jumbo Frame Enable** When this bit is set, the MAC allows Jumbo frames of 9,018 bytes (9,022 bytes for VLAN tagged frames) without reporting a giant frame error in the receive frame status.

Field	Bits	Type	Description
BE	21	r	**Frame Burst Enable** When this bit is set, the MAC allows frame bursting during transmission in the GMII half-duplex mode. This bit is reserved (and RO) in the 10/100 Mbps only or full-duplex-only configurations.
JD	22	rw	**Jabber Disable** When this bit is set, the MAC disables the jabber timer on the transmitter. The MAC can transfer frames of up to 16,384 bytes. When this bit is reset, the MAC cuts off the transmitter if the application sends out more than 2,048 bytes of data (10,240 if JE is set high) during transmission.
WD	23	rw	**Watchdog Disable** When this bit is set, the MAC disables the watchdog timer on the receiver. The MAC can receive frames of up to 16,384 bytes. When this bit is reset, the MAC does not allow more than 2,048 bytes (10,240 if JE is set high) of the frame being received. The MAC cuts off any bytes received after 2,048 bytes.
TC	24	r	**Transmit Configuration in RGMII, SGMII, or SMII** When set, this bit enables the transmission of duplex mode, link speed, and link up or down information to the PHY in the RGMII, SMII, or SGMII port. When this bit is reset, no such information is driven to the PHY. This bit is reserved (and RO) if the RGMII, SMII, or SGMII PHY port is not selected during core configuration.
CST	25	rw	**CRC Stripping of Type Frames** When set, the last 4 bytes (FCS) of all frames of Ether type (type field greater than 0x0600) are stripped and dropped before forwarding the frame to the application. This function is not valid when the IP Checksum Engine (Type 1) is enabled in the MAC receiver.
SFTERR	26	r	**SMII Force Transmit Error** When set, this bit indicates to the PHY to force a transmit error in the SMII frame being transmitted. This bit is reserved if the SMII PHY port is not selected during core configuration.

Field	Bits	Type	Description
TWOKPE	27	rw	**IEEE 802.3as support for 2K packets Enable** When set, the MAC considers all frames, with up to 2,000 bytes length, as normal packets. When Bit 20 (Jumbo Enable) is not set, the MAC considers all received frames of size more than 2K bytes as Giant frames. When this bit is reset and Bit 20 (Jumbo Enable) is not set, the MAC considers all received frames of size more than 1,518 bytes (1,522 bytes for tagged) as Giant frames. When Bit 20 (Jumbo Enable) is set, setting this bit has no effect on Giant Frame status.

Field	Bits	Type	Description
SARC	[30:28]	r	**Source Address Insertion or Replacement Control** This field controls the source address insertion or replacement for all transmitted frames. Bit 30 specifies which MAC Address register (0 or 1) is used for source address insertion or replacement based on the values of Bits [29:28]: * 2'b0x: The input signals mti_sa_ctrl_i and ati_sa_ctrl_i control the SA field generation. * 2'b10: - If Bit 30 is set to 0, the MAC inserts the content of the MAC Address 0 registers (registers 16 and 17) in the SA field of all transmitted frames. - If Bit 30 is set to 1 and the Enable MAC Address Register 1 option is selected during core configuration, the MAC inserts the content of the MAC Address 1 registers (registers 18 and 19) in the SA field of all transmitted frames. * 2'b11: - If Bit 30 is set to 0, the MAC replaces the content of the MAC Address 0 registers (registers 16 and 17) in the SA field of all transmitted frames. - If Bit 30 is set to 1 and the Enable MAC Address Register 1 option is selected during core configuration, the MAC replaces the content of the MAC Address 1 registers (registers 18 and 19) in the SA field of all transmitted frames. Note: - Changes to this field take effect only on the start of a frame. If you write this register field when a frame is being transmitted, only the subsequent frame can use the updated value, that is, the current frame does not use the updated value. - These bits are reserved and RO when the Enable SA, VLAN, and CRC Insertion on TX feature is not selected during core configuration.
RESERVED_31	31	r	**RESERVED_31**

32-bit Register - MAC_Frame_Filter

The MAC Frame Filter register contains the filter controls for receiving frames. Some of the controls from this register go to the address check block of the MAC, which performs the first level of address filtering. The second level of filtering is performed on the incoming frame, based on other controls such as Pass Bad Frames and Pass Control Frames.

ETH_MAC_FRAME_FILTER
Register 1 - MAC Frame Filter **(1004_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RA	RESERVED_30_22									DNTU	IPFE	RESERVED_19_17			VTFE
rw	r									r	r	r			rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED_15_11					HPF	SAF	SAIF	PCF		DBF	PM	DAIF	HMC	HUC	PR
r					rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PR	0	rw	**Promiscuous Mode** When this bit is set, the Address Filter module passes all incoming frames regardless of its destination or source address. The SA or DA Filter Fails status bits of the Receive Status Word are always cleared when PR is set.
HUC	1	rw	**Hash Unicast** When set, MAC performs destination address filtering of unicast frames according to the hash table. When reset, the MAC performs a perfect destination address filtering for unicast frames, that is, it compares the DA field with the values programmed in DA registers. If Hash Filter is not selected during core configuration, this bit is reserved (and RO).

Field	Bits	Type	Description
HMC	2	rw	**Hash Multicast** When set, MAC performs destination address filtering of received multicast frames according to the hash table. When reset, the MAC performs a perfect destination address filtering for multicast frames, that is, it compares the DA field with the values programmed in DA registers. If Hash Filter is not selected during core configuration, this bit is reserved (and RO).
DAIF	3	rw	**DA Inverse Filtering** When this bit is set, the Address Check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast frames. When reset, normal filtering of frames is performed.
PM	4	rw	**Pass All Multicast** When set, this bit indicates that all received frames with a multicast destination address (first bit in the destination address field is '1') are passed. When reset, filtering of multicast frame depends on HMC bit.
DBF	5	rw	**Disable Broadcast Frames** When this bit is set, the AFM module filters all incoming broadcast frames. In addition, it overrides all other filter settings. When this bit is reset, the AFM module passes all received broadcast frames.

Field	Bits	Type	Description
PCF	[7:6]	rw	**Pass Control Frames** These bits control the forwarding of all control frames (including unicast and multicast PAUSE frames). * 00: MAC filters all control frames from reaching the application. * 01: MAC forwards all control frames except PAUSE control frames to application even if they fail the Address filter. * 10: MAC forwards all control frames to application even if they fail the Address Filter. * 11: MAC forwards control frames that pass the Address Filter. The following conditions should be true for the PAUSE control frames processing: * Condition 1: The MAC is in the full-duplex mode and flow control is enabled by setting Bit 2 (RFE) of Register 6 (Flow Control Register) to 1. * Condition 2: The destination address (DA) of the received frame matches the special multicast address or the MAC Address 0 when Bit 3 (UP) of the Register 6 (Flow Control Register) is set. * Condition 3: The Type field of the received frame is 0x8808 and the OPCODE field is 0x0001. Note: This field should be set to 01 only when the Condition 1 is true, that is, the MAC is programmed to operate in the full-duplex mode and the RFE bit is enabled. Otherwise, the PAUSE frame filtering may be inconsistent. When Condition 1 is false, the PAUSE frames are considered as generic control frames. Therefore, to pass all control frames (including PAUSE control frames) when the full-duplex mode and flow control is not enabled, you should set the PCF field to 10 or 11 (as required by the application).

Field	Bits	Type	Description
SAIF	8	rw	**SA Inverse Filtering** When this bit is set, the Address Check block operates in inverse filtering mode for the SA address comparison. The frames whose SA matches the SA registers are marked as failing the SA Address filter. When this bit is reset, frames whose SA does not match the SA registers are marked as failing the SA Address filter.
SAF	9	rw	**Source Address Filter Enable** When this bit is set, the MAC compares the SA field of the received frames with the values programmed in the enabled SA registers. If the comparison matches, then the SA Match bit of RxStatus Word is set high. When this bit is set high and the SA filter fails, the MAC drops the frame. When this bit is reset, the MAC forwards the received frame to the application and with the updated SA Match bit of the RxStatus depending on the SA address comparison.
HPF	10	rw	**Hash or Perfect Filter** When this bit is set, it configures the address filter to pass a frame if it matches either the perfect filtering or the hash filtering as set by the HMC or HUC bits. When this bit is low and the HUC or HMC bit is set, the frame is passed only if it matches the Hash filter. This bit is reserved (and RO) if the Hash filter is not selected during core configuration.
RESERVED_15_11	[15:11]	r	**RESERVED_15_11**
VTFE	16	rw	**VLAN Tag Filter Enable** When set, this bit enables the MAC to drop VLAN tagged frames that do not match the VLAN Tag comparison. When reset, the MAC forwards all frames irrespective of the match status of the VLAN Tag.
RESERVED_19_17	[19:17]	r	**RESERVED_19_17**

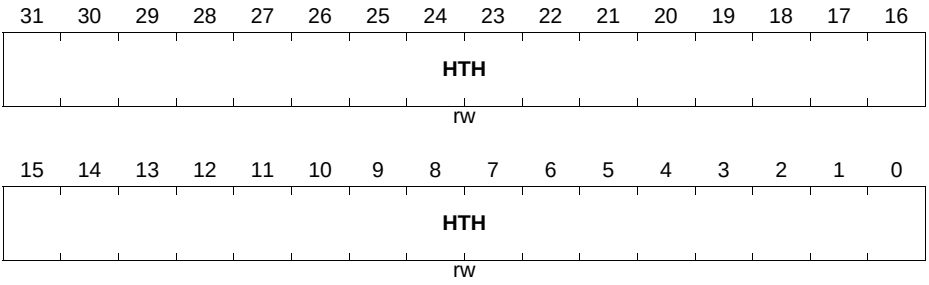
Field	Bits	Type	Description
IPFE	20	r	**Layer 3 and Layer 4 Filter Enable** When set, this bit enables the MAC to drop frames that do not match the enabled Layer 3 and Layer 4 filters. If Layer 3 or Layer 4 filters are not enabled for matching, this bit does not have any effect. When reset, the MAC forwards all frames irrespective of the match status of the Layer 3 and Layer 4 fields. If the Layer 3 and Layer 4 Filtering feature is not selected during core configuration, this bit is reserved (RO with default value).
DNTU	21	r	**Drop non-TCP/UDP over IP Frames** When set, this bit enables the MAC to drop the non-TCP or UDP over IP frames. The MAC forward only those frames that are processed by the Layer 4 filter. When reset, this bit enables the MAC to forward all non-TCP or UDP over IP frames. If the Layer 3 and Layer 4 Filtering feature is not selected during core configuration, this bit is reserved (RO with default value).
RESERVED_30_22	[30:22]	r	**RESERVED_30_22**
RA	31	rw	**Receive All** When this bit is set, the MAC Receiver module passes all received frames, irrespective of whether they pass the address filter or not, to the Application. The result of the SA or DA filtering is updated (pass or fail) in the corresponding bits in the Receive Status Word. When this bit is reset, the Receiver module passes only those frames to the Application that pass the SA or DA address filter.

32-bit Register - Hash_Table_High

The 64-bit Hash table is used for group address filtering. For hash filtering, the contents of the destination address in the incoming frame is passed through the CRC logic, and the upper 6 bits of the CRC register are used to index the contents of the Hash table. The most significant bit determines the register to be used (Hash Table High or Hash Table Low), and the other 5 bits determine which bit within the register. A hash value of 5b'00000 selects Bit 0 of the selected register, and a value of 5b'11111 selects Bit 31 of the selected register. The hash value of the destination address is calculated in the following way: 1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
 2. Perform bitwise reversal for the value obtained in Step 1.
 3. Take the upper 6 bits from the value obtained in Step 2.
 For example, if the DA of the incoming frame is received as 0x1F52419CB6AF (0x1F is the first byte received on GMII interface), then the internally calculated 6-bit Hash value is 0x2C and Bit 12 of Hash Table High register is checked for filtering. If the DA of the incoming frame is received as 0xA00A98000045, then the calculated 6-bit Hash value is 0x07 and Bit 7 of Hash Table Low register is checked for filtering. Note: To help you program the hash table, a sample C routine that generates a DA's 6-bit hash is included in the /sample_codes/ directory of your workspace. If the corresponding bit value of the register is 1'b1, the frame is accepted. Otherwise, it is rejected. If the PM (Pass All Multicast) bit is set in Register 1, then all multicast frames are accepted regardless of the multicast hash values. If the Hash Table register is configured to be double-synchronized to the (G)MII clock domain, the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the Hash Table High or Low registers are written. Consecutive writes to these register should be performed only after at least four clock cycles in the destination clock domain when double-synchronization is enabled. The Hash Table High register contains the higher 32 bits of the Hash table.

ETH_HASH_TABLE_HIGH

Register 2 - Hash Table High Register (1008_H) **Reset Value: 0000 0000_H**



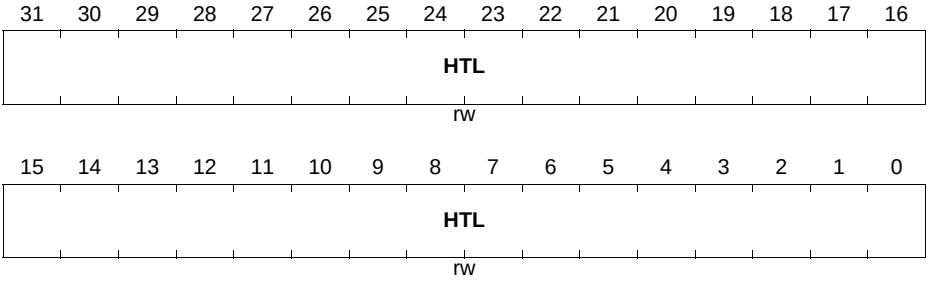
Field	Bits	Type	Description
HTH	[31:0]	rw	**Hash Table High** This field contains the upper 32 bits of the Hash table.

32-bit Register - Hash_Table_Low

The Hash Table Low register contains the lower 32 bits of the Hash table. Both Register 2 and Register 3 are reserved if the Hash Filter Function is disabled or the 128-bit or 256-bit Hash Table is selected during core configuration.

ETH_HASH_TABLE_LOW

Register 3 - Hash Table Low Register ($100C_H$) **Reset Value: $0000\ 0000_H$**

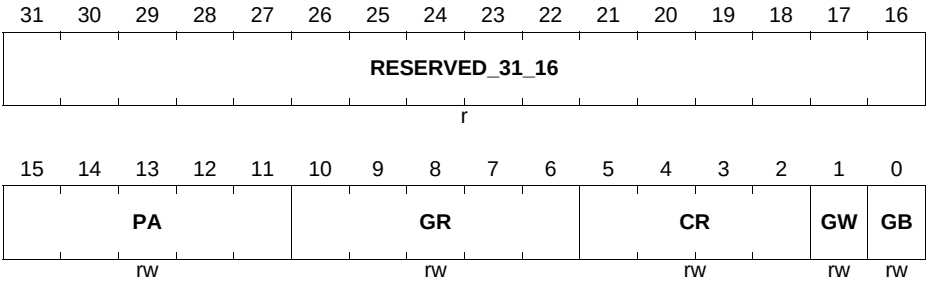


Field	Bits	Type	Description
HTL	[31:0]	rw	**Hash Table Low** This field contains the lower 32 bits of the Hash table.

32-bit Register - GMII_Address

The GMII Address register controls the management cycles to the external PHY through the management interface.

ETH_GMII_ADDRESS
Register 4 - GMII Address Register (1010$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
GB	0	rw	**GMII Busy** This bit should read logic 0 before writing to Register 4 and Register 5. During a PHY or RevMII register access, the software sets this bit to 1'b1 to indicate that a Read or Write access is in progress. The Register 5 is invalid until this bit is cleared by the MAC. Therefore, Register 5 (GMII Data) should be kept valid until the MAC clears this bit during a PHY Write operation. Similarly for a read operation, the contents of Register 5 are not valid until this bit is cleared. The subsequent read or write operation should happen only after the previous operation is complete. Because there is no acknowledgment from the PHY to MAC after a read or write operation is completed, there is no change in the functionality of this bit even when the PHY is not present.
GW	1	rw	**GMII Write** When set, this bit indicates to the PHY or RevMII that this is a Write operation using the GMII Data register. If this bit is not set, it indicates that this is a Read operation, that is, placing the data in the GMII Data register.

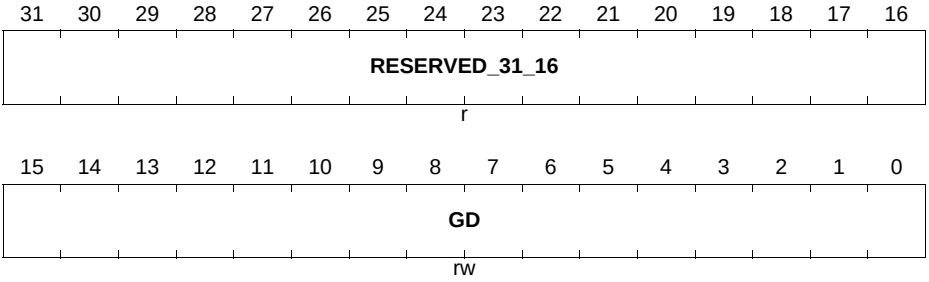
Field	Bits	Type	Description
CR	[5:2]	rw	**CSR Clock Range** The CSR Clock Range selection determines the frequency of the MDC clock according to the clk_csr_i frequency used in your design. The suggested range of clk_csr_i frequency applicable for each value (when Bit[5] = 0) ensures that the MDC clock is approximately between the frequency range 1.0 MHz - 2.5 MHz. - 0000: The frequency of the clk_csr_i clock is 60-100 MHz and the MDC clock is clk_csr_i/42. - 0001: The frequency of the clk_csr_i clock is 100-150 MHz and the MDC clock is clk_csr_i/62. - 0010: The frequency of the clk_csr_i clock is 20-35 MHz and the MDC clock is clk_csr_i/16. - 0011: The frequency of the clk_csr_i clock is 35-60 MHz and the MDC clock is clk_csr_i/26. - 0100: The frequency of the clk_csr_i clock is 150-250 MHz and the MDC clock is clk_csr_i/102. - 0100: The frequency of the clk_csr_i clock is 250-300 MHz and the MDC clock is clk_csr_i/124. - 0110 and 0111: Reserved When Bit 5 is set, you can achieve MDC clock of frequency higher than the IEEE 802.3 specified frequency limit of 2.5 MHz and program a clock divider of lower value. For example, when clk_csr_i is of 100 MHz frequency and you program these bits as 1010, then the resultant MDC clock is of 12.5 MHz which is outside the limit of IEEE 802.3 specified range. Program the following values only if the interfacing chips support faster MDC clocks: - 1000: clk_csr_i/4 - 1001: clk_csr_i/6 - 1010: clk_csr_i/8 - 1011: clk_csr_i/10 - 1100: clk_csr_i/12 - 1101: clk_csr_i/14 - 1110: clk_csr_i/16 - 1111: clk_csr_i/18 These bits are not used for accessing RevMII. These bits are read-only if the RevMII interface is selected as single PHY interface.

Field	Bits	Type	Description
GR	[10:6]	rw	**GMII Register** These bits select the desired GMII register in the selected PHY device. For RevMII, these bits select the desired CSR register in the RevMII Registers set.
PA	[15:11]	rw	**Physical Layer Address** This field indicates which of the 32 possible PHY devices are being accessed. For RevMII, this field gives the PHY Address of the RevMII module.
RESERVED_31_16	[31:16]	r	**RESERVED_31_16**

32-bit Register - GMII_Data

The GMII Data register stores Write data to be written to the PHY register located at the address specified in Register 4 (GMII Address Register). This register also stores the Read data from the PHY register located at the address specified by Register 4.

ETH_GMII_DATA
Register 5 - GMII Data Register (1014_H) Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
GD	[15:0]	rw	**GMII Data** This field contains the 16-bit data value read from the PHY or RevMII after a Management Read operation or the 16-bit data value to be written to the PHY or RevMII before a Management Write operation.
RESERVED_31_16	[31:16]	r	**RESERVED_31_16**

32-bit Register - Flow_Control

The Flow Control register controls the generation and reception of the Control (Pause Command) frames by the MAC's Flow control module. A Write to a register with the Busy bit set to '1' triggers the Flow Control block to generate a Pause Control frame. The fields of the control frame are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control frame. The Busy bit remains set until the control frame is transferred onto the cable. The Host must make sure that the Busy bit is cleared before writing to the register.

ETH_FLOW_CONTROL
Register 6 - Flow Control Register (1018$_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PT															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED_15_8								DZPQ	RESERVED_6	PLT		UP	RFE	TFE	FCA_BPA
r								rw	r	rw		rw	rw	rw	rw

Field	Bits	Type	Description
FCA_BPA	0	rw	**Flow Control Busy or Backpressure Activate** This bit initiates a Pause Control frame in the full-duplex mode and activates the backpressure function in the half-duplex mode if the TFE bit is set. In the full-duplex mode, this bit should be read as 1'b0 before writing to the Flow Control register. To initiate a Pause control frame, the Application must set this bit to 1'b1. During a transfer of the Control Frame, this bit continues to be set to signify that a frame transmission is in progress. After the completion of Pause control frame transmission, the MAC resets this bit to 1'b0. The Flow Control register should not be written to until this bit is cleared. In the half-duplex mode, when this bit is set (and TFE is set), then backpressure is asserted by the MAC. During backpressure, when the MAC receives a new frame, the transmitter starts sending a JAM pattern resulting in a collision. This control register bit is logically ORed with the mti_flowctrl_i input signal for the backpressure function. When the MAC is configured for the full-duplex mode, the BPA is automatically disabled.
TFE	1	rw	**Transmit Flow Control Enable** In the full-duplex mode, when this bit is set, the MAC enables the flow control operation to transmit Pause frames. When this bit is reset, the flow control operation in the MAC is disabled, and the MAC does not transmit any Pause frames. In half-duplex mode, when this bit is set, the MAC enables the back-pressure operation. When this bit is reset, the back-pressure feature is disabled.
RFE	2	rw	**Receive Flow Control Enable** When this bit is set, the MAC decodes the received Pause frame and disables its transmitter for a specified (Pause) time. When this bit is reset, the decode function of the Pause frame is disabled.

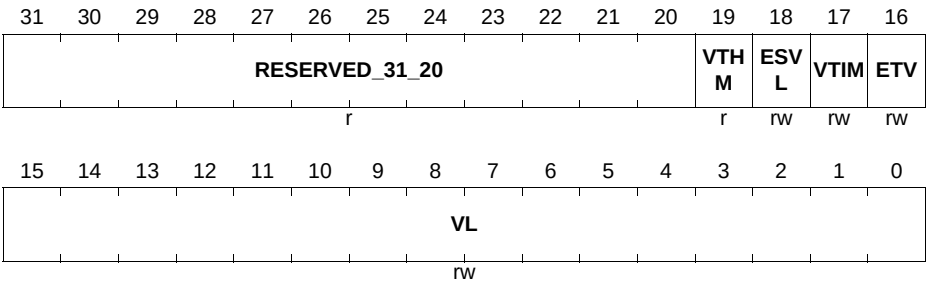
Field	Bits	Type	Description
UP	3	rw	**Unicast Pause Frame Detect** When this bit is set, then in addition to the detecting Pause frames with the unique multicast address, the MAC detects the Pause frames with the station's unicast address specified in the MAC Address0 High Register and MAC Address0 Low Register. When this bit is reset, the MAC detects only a Pause frame with the unique multicast address specified in the 802.3x standard.
PLT	[5:4]	rw	**Pause Low Threshold** This field configures the threshold of the PAUSE timer at which the input flow control signal mti_flowctrl_i (or sbd_flowctrl_i) is checked for automatic retransmission of PAUSE Frame. The threshold values should be always less than the Pause Time configured in Bits[31:16]. For example, if PT = 100H (256 slot-times), and PLT = 01, then a second PAUSE frame is automatically transmitted if the mti_flowctrl_i signal is asserted at 228 (256 - 28) slot times after the first PAUSE frame is transmitted. The following list provides the threshold values for different values: - 00: The threshold is Pause time minus 4 slot times (PT - 4 slot times). - 01: The threshold is Pause time minus 28 slot times (PT - 28 slot times). - 10: The threshold is Pause time minus 144 slot times (PT - 144 slot times). - 11: The threshold is Pause time minus 256 slot times (PT - 256 slot times). The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the GMII or MII interface.
RESERVED_6	6	r	**RESERVED_6**

Field	Bits	Type	Description
DZPQ	7	rw	**Disable Zero-Quanta Pause** When this bit is set, it disables the automatic generation of the Zero-Quanta Pause Control frames on the de-assertion of the flow-control signal from the FIFO layer (MTL or external sideband flow control signal sbd_flowctrl_i/mti_flowctrl_i). When this bit is reset, normal operation with automatic Zero-Quanta Pause Control frame generation is enabled.
RESERVED_15_8	[15:8]	r	**RESERVED_15_8**
PT	[31:16]	rw	**Pause Time** This field holds the value to be used in the Pause Time field in the transmit control frame. If the Pause Time bits is configured to be double-synchronized to the (G)MII clock domain, then consecutive writes to this register should be performed only after at least four clock cycles in the destination clock domain.

32-bit Register - VLAN_Tag

The VLAN Tag register contains the IEEE 802.1Q VLAN Tag to identify the VLAN frames. The MAC compares the 13th and 14th bytes of the receiving frame (Length/Type) with 16'h8100, and the following two bytes are compared with the VLAN tag. If a match occurs, the MAC sets the received VLAN bit in the receive frame status. The legal length of the frame is increased from 1,518 bytes to 1,522 bytes. If the VLAN Tag register is configured to be double-synchronized to the (G)MII clock domain, then consecutive writes to these register should be performed only after at least four clock cycles in the destination clock domain.

ETH_VLAN_TAG
Register 7 - VLAN Tag Register (101C$_H$) Reset Value: 0000 0000$_H$



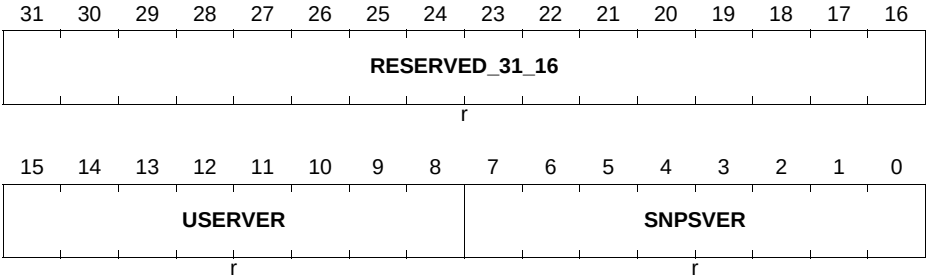
Field	Bits	Type	Description
VL	[15:0]	rw	**VLAN Tag Identifier for Receive Frames** This field contains the 802.1Q VLAN tag to identify the VLAN frames and is compared to the 15th and 16th bytes of the frames being received for VLAN frames. The following list describes the bits of this field: * Bits [15:13]: User Priority * Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI) * Bits[11:0]: VLAN tag's VLAN Identifier (VID) field When the ETV bit is set, only the VID (Bits[11:0]) is used for comparison. If VL (VL[11:0] if ETV is set) is all zeros, the MAC does not check the fifteenth and 16th bytes for VLAN tag comparison, and declares all frames with a Type field value of 0x8100 or 0x88a8 as VLAN frames.

Field	Bits	Type	Description
ETV	16	rw	**Enable 12-Bit VLAN Tag Comparison** When this bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits [11:0] of VLAN tag are compared with the corresponding field in the received VLAN-tagged frame. Similarly, when enabled, only 12 bits of the VLAN tag in the received frame are used for hash-based VLAN filtering. When this bit is reset, all 16 bits of the 15th and 16th bytes of the received VLAN frame are used for comparison and VLAN hash filtering.
VTIM	17	rw	**VLAN Tag Inverse Match Enable** When set, this bit enables the VLAN Tag inverse matching. The frames that do not have matching VLAN Tag are marked as matched. When reset, this bit enables the VLAN Tag perfect matching. The frames with matched VLAN Tag are marked as matched.
ESVL	18	rw	**Enable S-VLAN** When this bit is set, the MAC transmitter and receiver also consider the S-VLAN (Type = 0x88A8) frames as valid VLAN tagged frames.
VTHM	19	r	**VLAN Tag Hash Table Match Enable** When set, the most significant four bits of the VLAN tag¡Çs CRC are used to index the content of Register 354 (VLAN Hash Table Register). A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the frame matched the VLAN hash table. When Bit 16 (ETV) is set, the CRC of the 12-bit VLAN Identifier (VID) is used for comparison whereas when ETV is reset, the CRC of the 16-bit VLAN tag is used for comparison. When reset, the VLAN Hash Match operation is not performed. If the VLAN Hash feature is not enabled during core configuration, this bit is reserved (RO with default value).
RESERVED_31_20	[31:20]	r	**RESERVED_31_20**

32-bit Register - Version

The Version registers identifies the version of the DWC_gmac. This register contains two bytes: one that Synopsys uses to identify the core release number, and the other that you set during core configuration.

ETH_VERSION
Register 8 - Version Register (1020_H) Reset Value: $0000\ 5537_H$

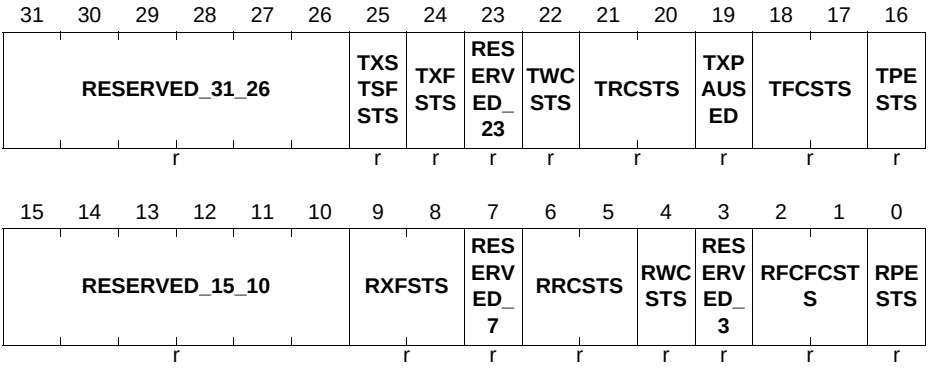


Field	Bits	Type	Description
SNPSVER	[7:0]	r	**SNPSVER** Synopsys-defined Version (3.7)
USERVER	[15:8]	r	**USERVER** User-defined Version (Configured with the coreConsultant)
RESERVED_31_16	[31:16]	r	**RESERVED_31_16**

32-bit Register - Debug

The Debug register gives the status of all main modules of the transmit and receive data-paths and the FIFOs. An all-zero status indicates that the MAC is in idle state (and FIFOs are empty) and no activity is going on in the data-paths. Note: The reset values, given for the Debug register, are valid only if the following clocks are present during the reset operation: * clk_csr_i, clk_app_i, hclk_i, or aclk_i * clk_tx_i * clk_rx_i

ETH_DEBUG
Register 9 - Debug Register (1024_H) Reset Value: $0000\ 0100_H$



Field	Bits	Type	Description
RPESTS	0	r	**MAC GMII or MII Receive Protocol Engine Status** When high, this bit indicates that the MAC GMII or MII receive protocol engine is actively receiving data and not in IDLE state.
RFCFCSTS	[2:1]	r	**MAC Receive Frame Controller FIFO Status** When high, this field indicates the active state of the small FIFO Read and Write controllers of the MAC Receive Frame Controller Module.
RESERVED_3	3	r	**RESERVED_3**
RWCSTS	4	r	**MTL Rx FIFO Write Controller Active Status** When high, this bit indicates that the MTL Rx FIFO Write Controller is active and is transferring a received frame to the FIFO.

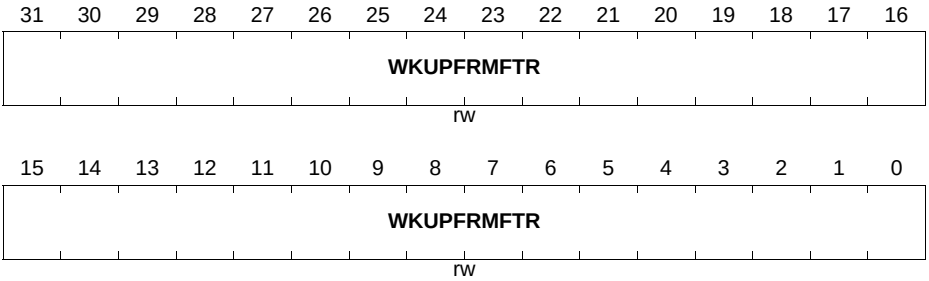
Field	Bits	Type	Description
RRCSTS	[6:5]	r	**MTL Rx FIFO Read Controller State** This field gives the state of the Rx FIFO read Controller: * 00: IDLE state * 01: Reading frame data * 10: Reading frame status (or timestamp) * 11: Flushing the frame data and status
RESERVED_7	7	r	**RESERVED_7**
RXFSTS	[9:8]	r	**MTL Rx FIFO Fill-level Status** This field gives the status of the fill-level of the Rx FIFO: * 00: Rx FIFO Empty * 01: Rx FIFO fill level is below the flow-control deactivate threshold * 10: Rx FIFO fill level is above the flow-control activate threshold * 11: Rx FIFO Full
RESERVED_15_10	[15:10]	r	**RESERVED_15_10**
TPESTS	16	r	**MAC GMII or MII Transmit Protocol Engine Status** When high, this bit indicates that the MAC GMII or MII transmit protocol engine is actively transmitting data and is not in the IDLE state.
TFCSTS	[18:17]	r	**MAC Transmit Frame Controller Status** This field indicates the state of the MAC Transmit Frame Controller module: * 00: IDLE state * 01: Waiting for Status of previous frame or IFG or backoff period to be over * 10: Generating and transmitting a PAUSE control frame (in the full-duplex mode) * 11: Transferring input frame for transmission
TXPAUSED	19	r	**MAC transmitter in PAUSE** When high, this bit indicates that the MAC transmitter is in the PAUSE condition (in the full-duplex only mode) and hence does not schedule any frame for transmission.

Field	Bits	Type	Description
TRCSTS	[21:20]	r	**MTL Tx FIFO Read Controller Status** This field indicates the state of the Tx FIFO Read Controller: * 00: IDLE state * 01: READ state (transferring data to MAC transmitter) * 10: Waiting for TxStatus from MAC transmitter * 11: Writing the received TxStatus or flushing the Tx FIFO
TWCSTS	22	r	**MTL Tx FIFO Write Controller Active Status** When high, this bit indicates that the MTL Tx FIFO Write Controller is active and transferring data to the Tx FIFO.
RESERVED_23	23	r	**RESERVED_23**
TXFSTS	24	r	**MTL Tx FIFO Not Empty Status** When high, this bit indicates that the MTL Tx FIFO is not empty and some data is left for transmission.
TXSTSFSTS	25	r	**MTL TxStatus FIFO Full Status** When high, this bit indicates that the MTL TxStatus FIFO is full. Therefore, the MTL cannot accept any more frames for transmission. This bit is reserved in the GMAC-AHB and GMAC-DMA configurations.
RESERVED_31_26	[31:26]	r	**RESERVED_31_26**

32-bit Register - Remote_Wake_Up_Frame_Filter

This is the address through which the application writes or reads the remote wake-up frame filter registers (wkupfmfilter_reg). The wkupfmfilter_reg register is a pointer to eight wkupfmfilter_reg registers. The wkupfmfilter_reg register is loaded by sequentially loading the eight register values. Eight sequential writes to this address (0x0028) writes all wkupfmfilter_reg registers. Similarly, eight sequential reads from this address (0x0028) read all wkupfmfilter_reg registers. This register contains the higher 16 bits of the seventh MAC address. This register is present only when you select the PMT module Remote Wake-up feature in coreConsultant.

ETH_REMOTE_WAKE_UP_FRAME_FILTER
Register 10 - Remote Wake-Up Frame Filter Register (1028_H) Reset Value: 0000 0000_H



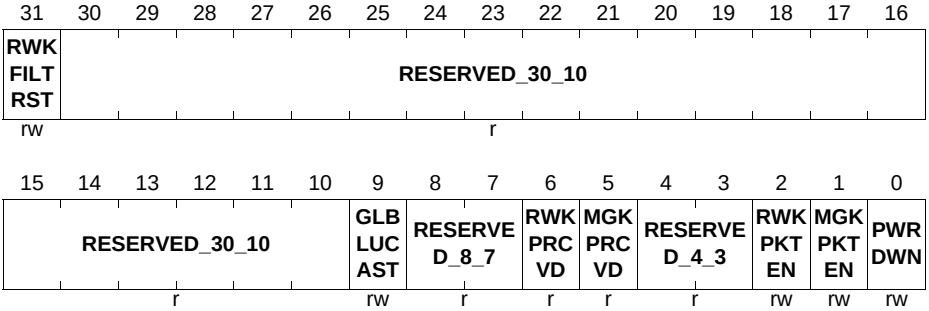
Field	Bits	Type	Description
WKUPFRMFTR	[31:0]	rw	**WKUPFRMFTR** Remote Wake-Up Frame Filter

32-bit Register - PMT_Control_Status

This register is present only when you select the PMT module in the coreConsultant.

ETH_PMT_CONTROL_STATUS

Register 11 - PMT Control and Status Register ($102C_H$) Reset Value: $0000\ 0000_H$



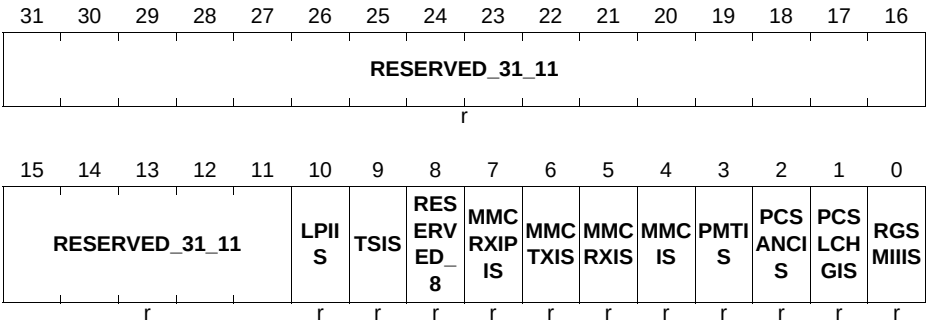
Field	Bits	Type	Description
PWRDWN	0	rw	**Power Down** When set, the MAC receiver drops all received frames until it receives the expected magic packet or wake-up frame. This bit is then self-cleared and the power-down mode is disabled. The Software can also clear this bit before the expected magic packet or wake-up frame is received. The frames, received by the MAC after this bit is cleared, are forwarded to the application. This bit must only be set when the Magic Packet Enable, Global Unicast, or Wake-Up Frame Enable bit is set high. Note: You can gate-off the CSR clock during the power-down mode. However, when the CSR clock is gated-off, you cannot perform any read or write operations on this register. Therefore, the Software cannot clear this bit.
MGKPKTEN	1	rw	**Magic Packet Enable** When set, enables generation of a power management event because of magic packet reception.
RWKPKTEN	2	rw	**Wake-Up Frame Enable** When set, enables generation of a power management event because of wake-up frame reception.

Field	Bits	Type	Description
RESERVED_4_3	[4:3]	r	**RESERVED_4_3**
MGKPRCVD	5	r	**Magic Packet Received** When set, this bit indicates that the power management event is generated because of the reception of a magic packet. This bit is cleared by a Read into this register.
RWKPRCVD	6	r	**Wake-Up Frame Received** When set, this bit indicates the power management event is generated because of the reception of a wake-up frame. This bit is cleared by a Read into this register.
RESERVED_8_7	[8:7]	r	**RESERVED_8_7**
GLBLUCAST	9	rw	**Global Unicast** When set, enables any unicast packet filtered by the MAC (DAF) address recognition to be a wake-up frame.
RESERVED_30_10	[30:10]	r	**RESERVED_30_10**
RWKFILTRST	31	rw	**Wake-Up Frame Filter Register Pointer Reset** When set, resets the remote wake-up frame filter register pointer to 3¡Çb000. It is automatically cleared after 1 clock cycle.

32-bit Register - Interrupt_Status

The Interrupt Status register identifies the events in the MAC that can generate interrupt. All interrupt events are generated only when the corresponding optional feature is selected during core configuration and enabled during operation. Therefore, these bits are reserved when the corresponding features are not present in the core.

ETH_INTERRUPT_STATUS
Register 14 - Interrupt Register **(1038_H)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RGSMIIIS	0	r	**RGMII or SMII Interrupt Status** This bit is set because of any change in value of the Link Status of RGMII or SMII interface (Bit 3 in Register 54 (SGMII/RGMII/SMII Status Register)). This bit is cleared when you perform a read operation on the SGMII/RGMII/SMII Status Register. This bit is valid only when you select the optional RGMII or SMII PHY interface during core configuration and operation.
PCSLCHGIS	1	r	**PCS Link Status Changed** This bit is set because of any change in Link Status in the TBI, RTBI, or SGMII PHY interface (Bit 2 in Register 49 (AN Status Register)). This bit is cleared when you perform a read operation on the AN Status register. This bit is valid only when you select the optional TBI, RTBI, or SGMII PHY interface during core configuration and operation.

Field	Bits	Type	Description
PCSANCIS	2	r	**PCS Auto-Negotiation Complete** This bit is set when the Auto-negotiation is completed in the TBI, RTBI, or SGMII PHY interface (Bit 5 in Register 49 (AN Status Register)). This bit is cleared when you perform a read operation to the AN Status register. This bit is valid only when you select the optional TBI, RTBI, or SGMII PHY interface during core configuration and operation.
PMTIS	3	r	**PMT Interrupt Status** This bit is set when a Magic packet or Wake-on-LAN frame is received in the power-down mode (see Bits 5 and 6 in the PMT Control and Status Register). This bit is cleared when both Bits[6:5] are cleared because of a read operation to the PMT Control and Status register. This bit is valid only when you select the optional PMT module during core configuration.
MMCIS	4	r	**MMC Interrupt Status** This bit is set high when any of the Bits [7:5] is set high and cleared only when all of these bits are low. This bit is valid only when you select the optional MMC module during core configuration.
MMCRXIS	5	r	**MMC Receive Interrupt Status** This bit is set high when an interrupt is generated in the MMC Receive Interrupt Register. This bit is cleared when all the bits in this interrupt register are cleared. This bit is valid only when you select the optional MMC module during core configuration.
MMCTXIS	6	r	**MMC Transmit Interrupt Status** This bit is set high when an interrupt is generated in the MMC Transmit Interrupt Register. This bit is cleared when all the bits in this interrupt register are cleared. This bit is valid only when you select the optional MMC module during core configuration.

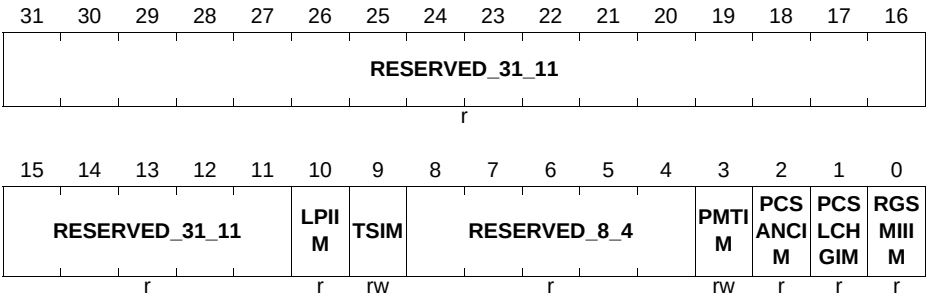
Field	Bits	Type	Description
MMCRXIPIS	7	r	**MMC Receive Checksum Offload Interrupt Status** This bit is set high when an interrupt is generated in the MMC Receive Checksum Offload Interrupt Register. This bit is cleared when all the bits in this interrupt register are cleared. This bit is valid only when you select the optional MMC module and Checksum Offload Engine (Type 2) during core configuration.
RESERVED_8	8	r	**RESERVED_8**
TSIS	9	r	**Timestamp Interrupt Status** When the Advanced Timestamp feature is enabled, this bit is set when any of the following conditions is true: * The system time value equals or exceeds the value specified in the Target Time High and Low registers. * There is an overflow in the seconds register. * The Auxiliary snapshot trigger is asserted. This bit is cleared on reading Bit 0 of the Register 458 (Timestamp Status Register). If default Timestamping is enabled, when set, this bit indicates that the system time value is equal to or exceeds the value specified in the Target Time registers. In this mode, this bit is cleared after the completion of the read of this bit. In all other modes, this bit is reserved.
LPIIS	10	r	**LPI Interrupt Status** When the Energy Efficient Ethernet feature is enabled, this bit is set for any LPI state entry or exit in the MAC Transmitter or Receiver. This bit is cleared on reading Bit 0 of Register 12 (LPI Control and Status Register). In all other modes, this bit is reserved.
RESERVED_31_11	[31:11]	r	**RESERVED_31_11**

32-bit Register - Interrupt_Mask

The Interrupt Mask Register bits enable you to mask the interrupt signal because of the corresponding event in the Interrupt Status Register. The interrupt signal is sbd_intr_o in the GMAC-AHB, GMAC-AXI, and GMAC-DMA configuration and mci_intr_o in the GMAC-MTL and GMAC-CORE configuration.

ETH_INTERRUPT_MASK

Register 15 - Interrupt Mask Register ($103C_H$) **Reset Value: $0000\ 0000_H$**



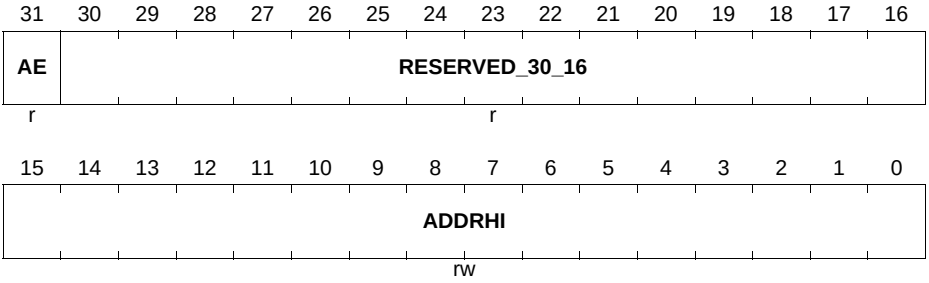
Field	Bits	Type	Description
RGSMIIIM	0	r	**RGMII or SMII Interrupt Mask** When set, this bit disables the assertion of the interrupt signal because of the setting of the RGMII or SMII Interrupt Status bit in Register 14 (Interrupt Status Register).
PCSLCHGIM	1	r	**PCS Link Status Interrupt Mask** When set, this bit disables the assertion of the interrupt signal because of the setting of the PCS Link-status changed bit in Register 14 (Interrupt Status Register).
PCSANCIM	2	r	**PCS AN Completion Interrupt Mask** When set, this bit disables the assertion of the interrupt signal because of the setting of PCS Auto-negotiation complete bit in Register 14 (Interrupt Status Register).
PMTIM	3	rw	**PMT Interrupt Mask** When set, this bit disables the assertion of the interrupt signal because of the setting of PMT Interrupt Status bit in Register 14 (Interrupt Status Register).
RESERVED_8_4	[8:4]	r	**RESERVED_8_4**

Field	Bits	Type	Description
TSIM	9	rw	**Timestamp Interrupt Mask** When set, this bit disables the assertion of the interrupt signal because of the setting of Timestamp Interrupt Status bit in Register 14 (Interrupt Status Register). This bit is valid only when IEEE1588 timestamping is enabled. In all other modes, this bit is reserved.
LPIIM	10	r	**LPI Interrupt Mask** When set, this bit disables the assertion of the interrupt signal because of the setting of the LPI Interrupt Status bit in Register 14 (Interrupt Status Register). This bit is valid only when you select the Energy Efficient Ethernet feature during core configuration. In all other modes, this bit is reserved.
RESERVED_31_11	[31:11]	r	**RESERVED_31_11**

32-bit Register - MAC_Address0_High

The MAC Address0 High register holds the upper 16 bits of the first 6-byte MAC address of the station. The first DA byte that is received on the (G)MII interface corresponds to the LS byte (Bits [7:0]) of the MAC Address Low register. For example, if 0x112233445566 is received (0x11 in lane 0 of the first column) on the (G)MII as the destination address, then the MacAddress0 Register [47:0] is compared with 0x665544332211. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address0 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS0_HIGH
Register 16 - MAC Address0 High Register (1040_H) **Reset Value: $8000\ FFFF_H$**



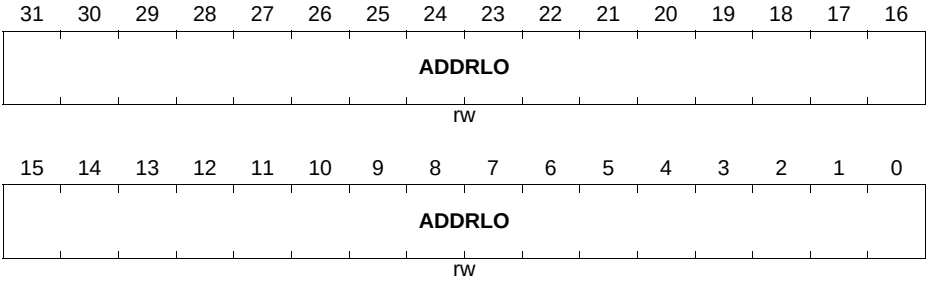
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address0 [47:32]** This field contains the upper 16 bits (47:32) of the first 6-byte MAC address. The MAC uses this field for filtering the received frames and inserting the MAC address in the Transmit Flow Control (PAUSE) Frames.
RESERVED_30_16	[30:16]	r	**RESERVED_30_16**
AE	31	r	**Address Enable** This bit is always set to 1.

32-bit Register - MAC_Address0_Low

The MAC Address0 Low register holds the lower 32 bits of the first 6-byte MAC address of the station.

ETH_MAC_ADDRESS0_LOW

Register 17 - MAC Address0 Low Register (1044_H) **Reset Value: $FFFF\ FFFF_H$**



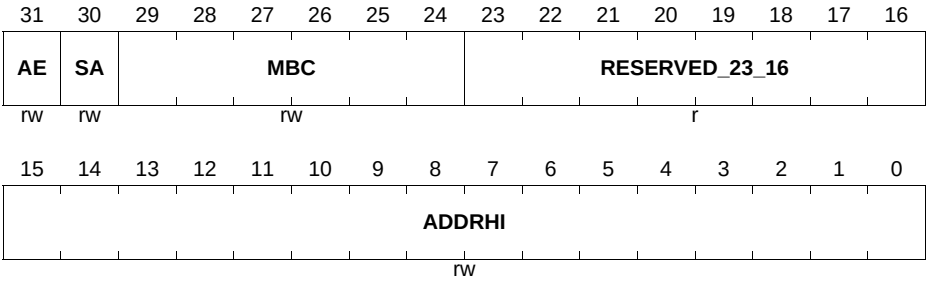
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address0 [31:0]** This field contains the lower 32 bits of the first 6-byte MAC address. This is used by the MAC for filtering the received frames and inserting the MAC address in the Transmit Flow Control (PAUSE) Frames.

32-bit Register - MAC_Address1_High

The MAC Address1 High register holds the upper 16 bits of the second 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address1 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS1_HIGH

Register 18 - MAC Address1 High Register (1048_H) **Reset Value: $0000\ FFFF_H$**



Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address1 [47:32]** This field contains the upper 16 bits (47:32) of the second 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**

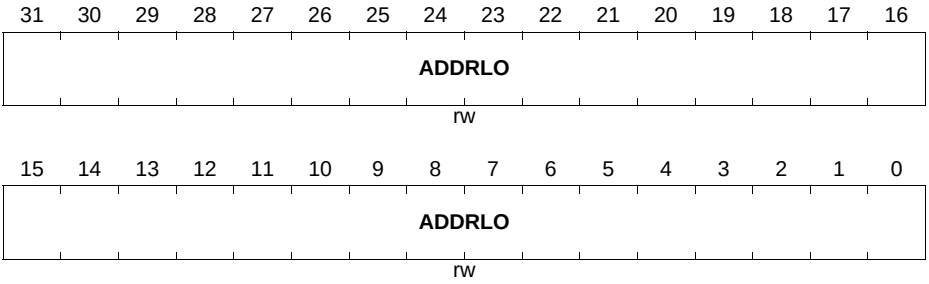
Field	Bits	Type	Description
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0] You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address.
SA	30	rw	**Source Address** When this bit is set, the MAC Address1[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address1[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the second MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address1_Low

The MAC Address1 Low register holds the lower 32 bits of the second 6-byte MAC address of the station.

ETH_MAC_ADDRESS1_LOW

Register 19 - MAC Address1 Low Register ($104C_H$) **Reset Value: $FFFF\ FFFF_H$**



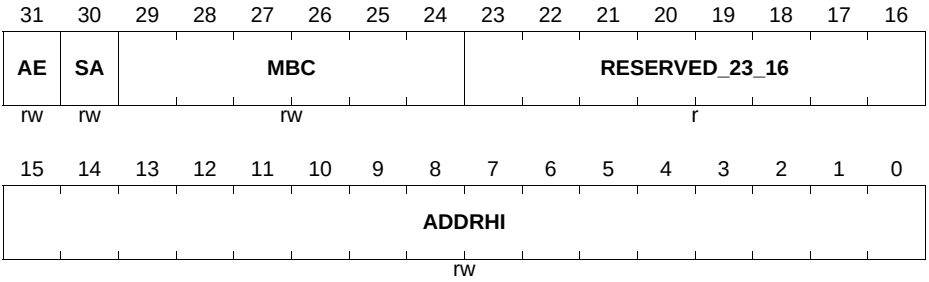
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address1 [31:0]** This field contains the lower 32 bits of the second 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address2_High

The MAC Address2 High register holds the upper 16 bits of the third 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address2 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address2 Low Register must be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS2_HIGH

Register 20 - MAC Address2 High Register (1050_H) **Reset Value: $0000\ FFFF_H$**



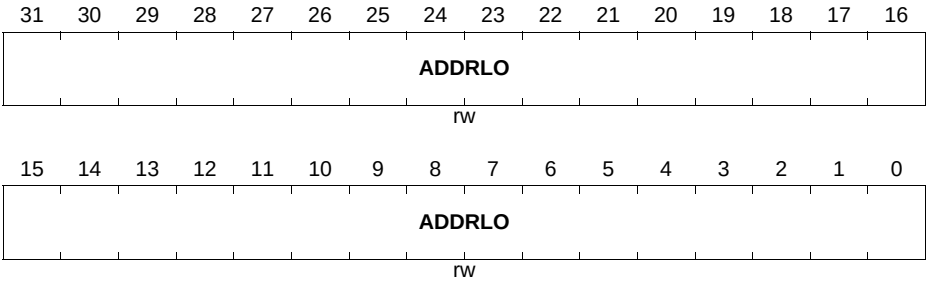
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address2 [47:32]** This field contains the upper 16 bits (47:32) of the third 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address2 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address2[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address2[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the third MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address2_Low

The MAC Address2 Low register holds the lower 32 bits of the third 6-byte MAC address of the station.

ETH_MAC_ADDRESS2_LOW
Register 21 - MAC Address2 Low Register (1054_H) **Reset Value: $FFFF\ FFFF_H$**



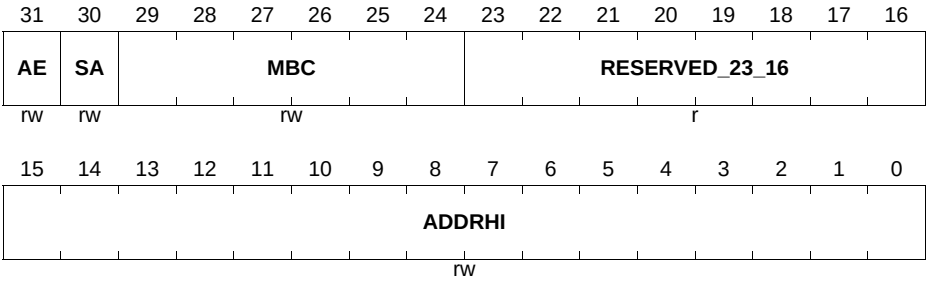
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address2 [31:0]** This field contains the lower 32 bits of the third 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address3_High

The MAC Address3 High register holds the upper 16 bits of the fourth 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address3 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address3 Low Register must be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS3_HIGH

Register 22 - MAC Address3 High Register (1058_H) **Reset Value: $0000\ FFFF_H$**



Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address3 [47:32]** This field contains the upper 16 bits (47:32) of the fourth 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address3 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

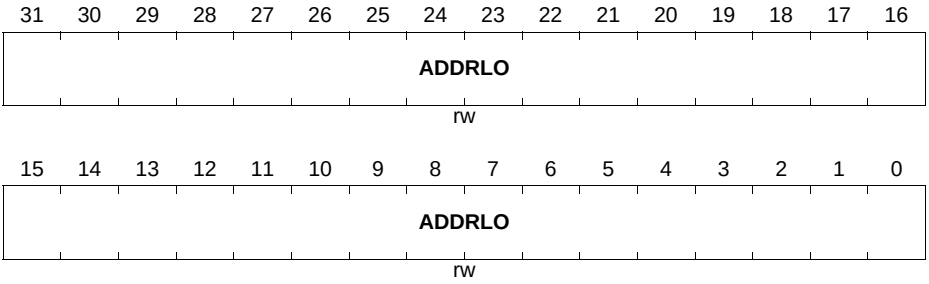
Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address3[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address3[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the fourth MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address3_Low

The MAC Address3 Low register holds the lower 32 bits of the fourth 6-byte MAC address of the station.

ETH_MAC_ADDRESS3_LOW

Register 23 - MAC Address3 Low Register ($105C_H$) **Reset Value: $FFFF\ FFFF_H$**



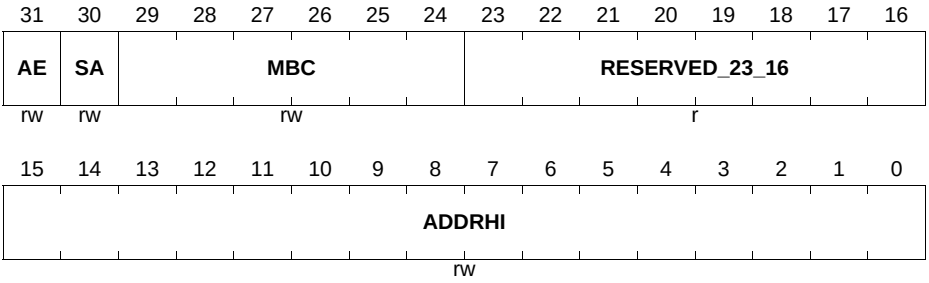
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address3 [31:0]** This field contains the lower 32 bits of the fourth 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address4_High

The MAC Address4 High register holds the upper 16 bits of the fifth 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address4 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address4 Low Register must be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS4_HIGH

Register 24 - MAC Address4 High Register (1060_H) **Reset Value: $0000\ FFFF_H$**



Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address4 [47:32]** This field contains the upper 16 bits (47:32) of the fifth 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address4 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

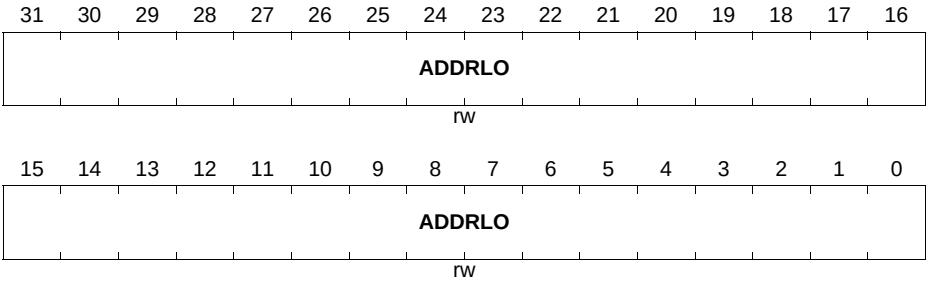
Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address4[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address4[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the fifth MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address4_Low

The MAC Address4 Low register holds the lower 32 bits of the fifth 6-byte MAC address of the station.

ETH_MAC_ADDRESS4_LOW

Register 25 - MAC Address4 Low Register (1064_H) **Reset Value: $FFFF\ FFFF_H$**



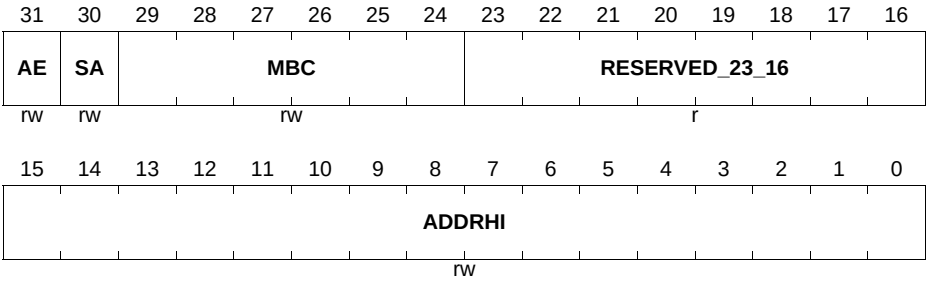
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address4 [31:0]** This field contains the lower 32 bits of the fifth 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address5_High

The MAC Address5 High register holds the upper 16 bits of the sixth 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address5 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address5 Low Register must be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS5_HIGH

Register 26 - MAC Address5 High Register (1068_H) **Reset Value: $0000\ FFFF_H$**



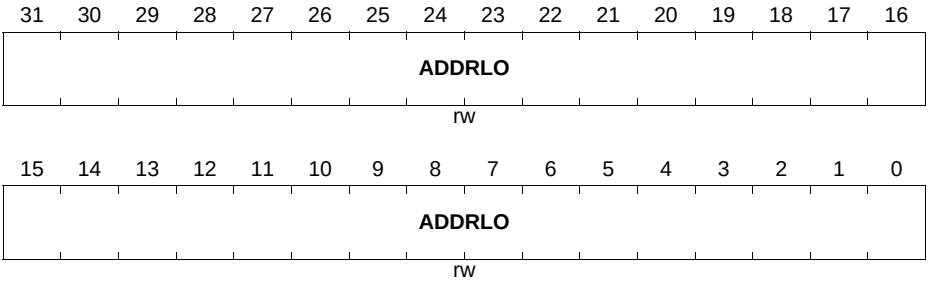
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address5 [47:32]** This field contains the upper 16 bits (47:32) of the sixth 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address5 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address5[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address5[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the sixth MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address5_Low

The MAC Address5 Low register holds the lower 32 bits of the sixth 6-byte MAC address of the station.

ETH_MAC_ADDRESS5_LOW
Register 27 - MAC Address5 Low Register ($106C_H$) **Reset Value: $FFFF\ FFFF_H$**



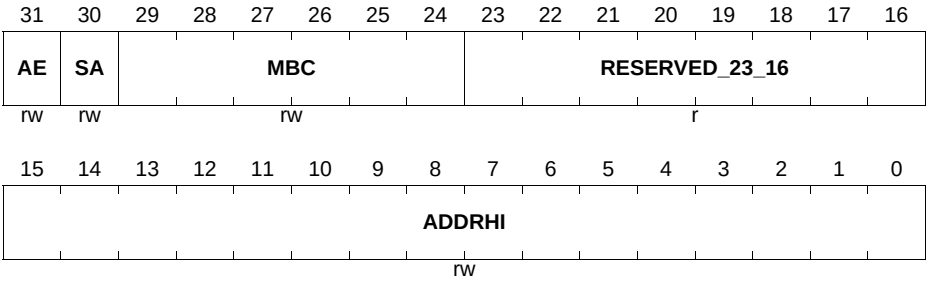
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address5 [31:0]** This field contains the lower 32 bits of the sixth 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address6_High

The MAC Address6 High register holds the upper 16 bits of the seventh 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address6 Low Register are written. For proper synchronization updates, the consecutive writes to this MAC Address6 Low Register should be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS6_HIGH

Register 28 - MAC Address6 High Register (1070_H) **Reset Value: $0000\ FFFF_H$**



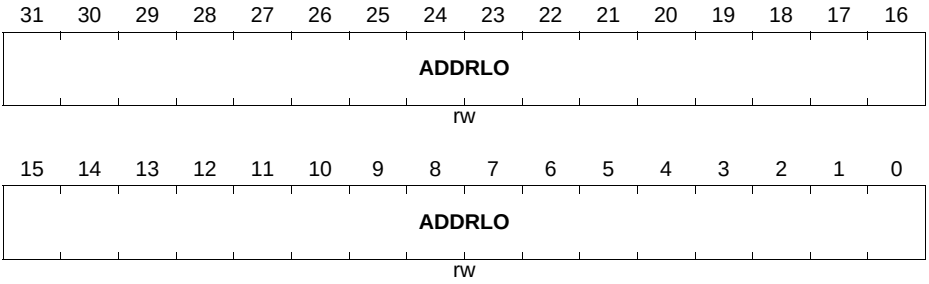
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address6 [47:32]** This field contains the upper 16 bits (47:32) of the seventh 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address6 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address6[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address6[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the seventh MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address6_Low

The MAC Address6 Low register holds the lower 32 bits of the seventh 6-byte MAC address of the station.

ETH_MAC_ADDRESS6_LOW
Register 29 - MAC Address6 Low Register (1074_H) **Reset Value: $FFFF\ FFFF_H$**



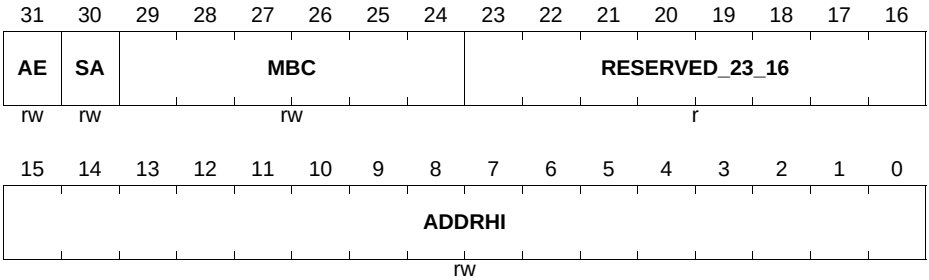
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address6 [31:0]** This field contains the lower 32 bits of the seventh 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address7_High

The MAC Address7 High register holds the upper 16 bits of the eighth 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address7 Low Register are written. For proper synchronization updates, the consecutive writes to this MAC Address7 Low Register should be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS7_HIGH

Register 30 - MAC Address7 High Register (1078_H) **Reset Value: $0000\ FFFF_H$**



Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address7 [47:32]** This field contains the upper 16 bits (47:32) of the eighth 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address6 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

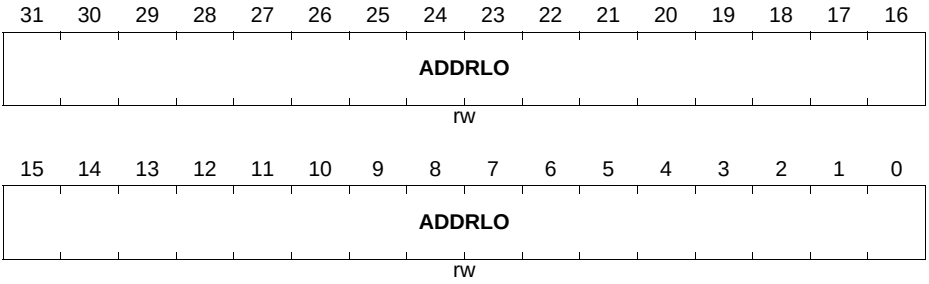
Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address7[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address7[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the eighth MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address7_Low

The MAC Address7 Low register holds the lower 32 bits of the eighth 6-byte MAC address of the station.

ETH_MAC_ADDRESS7_LOW

Register 31 - MAC Address7 Low Register ($107C_H$) **Reset Value: $FFFF\ FFFF_H$**



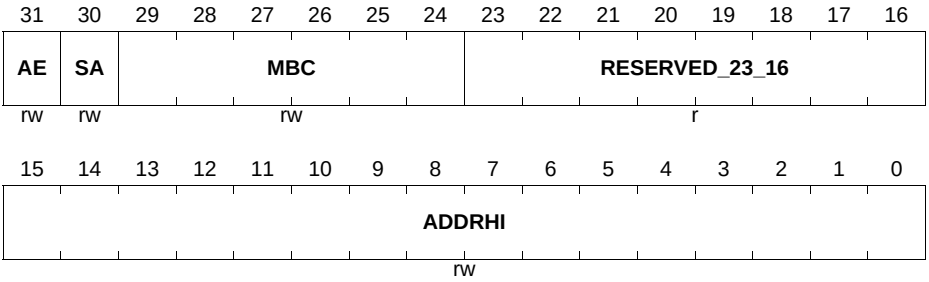
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address7 [31:0]** This field contains the lower 32 bits of the eighth 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address8_High

The MAC Address8 High register holds the upper 16 bits of the nineth 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address8 Low Register are written. For proper synchronization updates, the consecutive writes to this MAC Address8 Low Register should be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS8_HIGH

Register 32 - MAC Address8 High Register (1080_H) **Reset Value: $0000\ FFFF_H$**



Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address8 [47:32]** This field contains the upper 16 bits (47:32) of the nineth 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address8 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

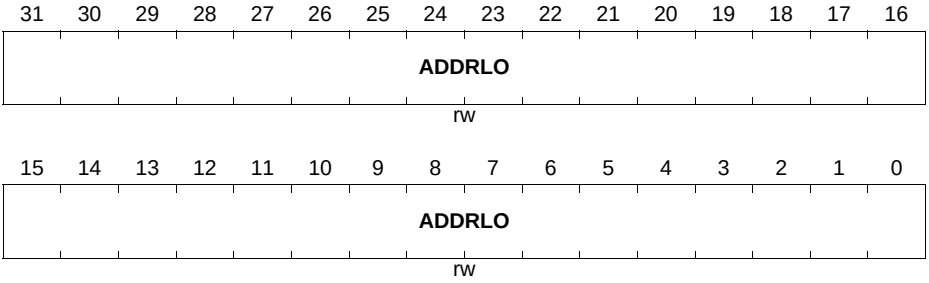
Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address8[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address8[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the nineth MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address8_Low

The MAC Address8 Low register holds the lower 32 bits of the nineth 6-byte MAC address of the station.

ETH_MAC_ADDRESS8_LOW

Register 33 - MAC Address8 Low Register (1084_H) **Reset Value: $FFFF\ FFFF_H$**



Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address8 [31:0]** This field contains the lower 32 bits of the nineth 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address9_High

The MAC Address9 High register holds the upper 16 bits of the tenth 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address9 Low Register are written. For proper synchronization updates, the consecutive writes to this MAC Address9 Low Register should be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS9_HIGH

Register 34 - MAC Address9 High Register (1088_H) **Reset Value: $0000\ FFFF_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	SA	MBC						RESERVED_23_16							
rw	rw	rw						r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRHI															
rw															

Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address9 [47:32]** This field contains the upper 16 bits (47:32) of the tenth 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address9 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

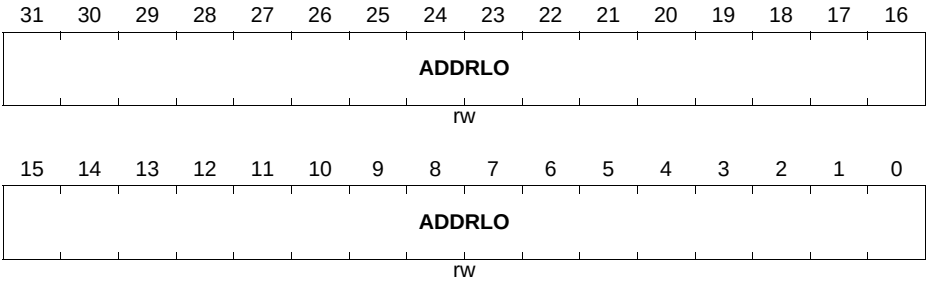
Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address9[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address9[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the tenth MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address9_Low

The MAC Address9 Low register holds the lower 32 bits of the tenth 6-byte MAC address of the station.

ETH_MAC_ADDRESS9_LOW

Register 35 - MAC Address9 Low Register ($108C_H$) **Reset Value: $FFFF\ FFFF_H$**



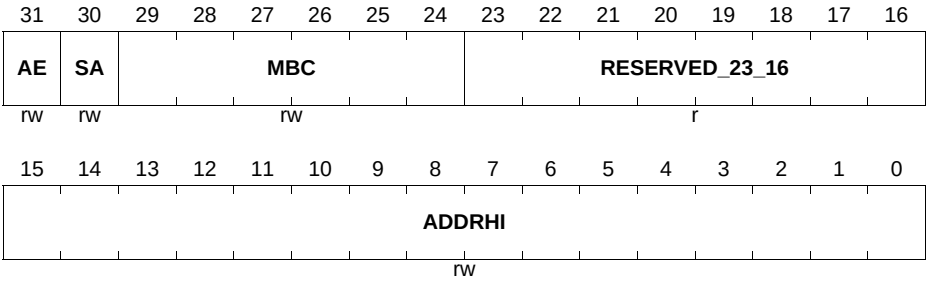
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address9 [31:0]** This field contains the lower 32 bits of the tenth 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address10_High

The MAC Address10 High register holds the upper 16 bits of the 11th 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address10 Low Register are written. For proper synchronization updates, the consecutive writes to this MAC Address10 Low Register should be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS10_HIGH

Register 36 - MAC Address10 High Register (1090_H) **Reset Value: $0000\ FFFF_H$**



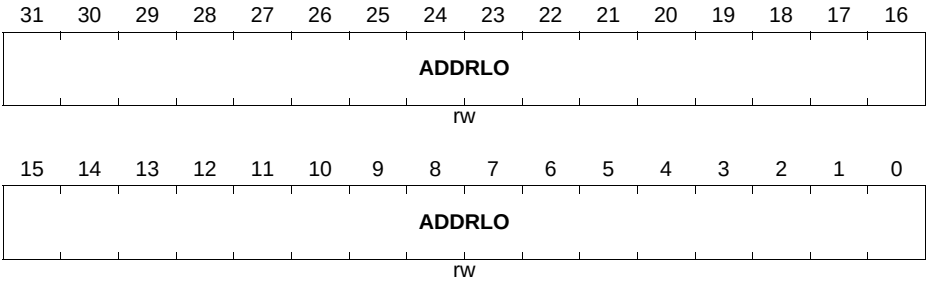
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address10 [47:32]** This field contains the upper 16 bits (47:32) of the 11th 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address10 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address10[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address10[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the 11th MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address10_Low

The MAC Address10 Low register holds the lower 32 bits of the 11th 6-byte MAC address of the station.

ETH_MAC_ADDRESS10_LOW
Register 37 - MAC Address10 Low Register (1094_H) **Reset Value: FFFF FFFF$_H$**

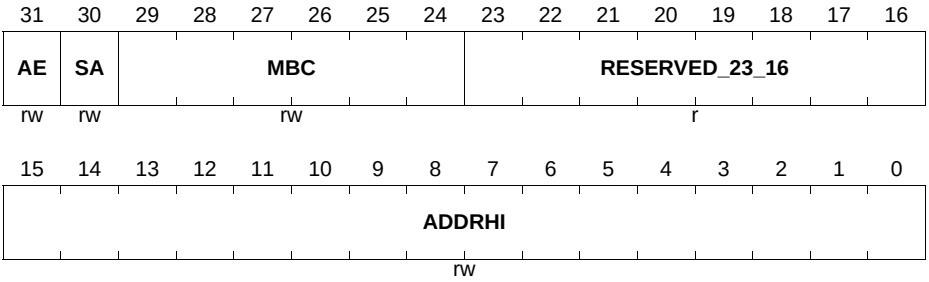


Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address10 [31:0]** This field contains the lower 32 bits of the 11th 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address11_High

The MAC Address11 High register holds the upper 16 bits of the 12th 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address11 Low Register are written. For proper synchronization updates, the consecutive writes to this MAC Address11 Low Register should be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS11_HIGH
Register 38 - MAC Address11 High Register (1098_H) **Reset Value: $0000\ FFFF_H$**



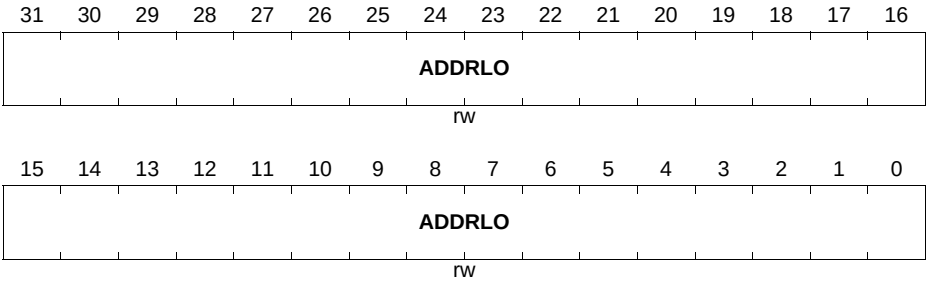
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address11 [47:32]** This field contains the upper 16 bits (47:32) of the 12th 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address11 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address11[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address11[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the twelfth MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address11_Low

The MAC Address11 Low register holds the lower 32 bits of the 12th 6-byte MAC address of the station.

ETH_MAC_ADDRESS11_LOW
Register 39 - MAC Address1 Low Register ($109C_H$) **Reset Value: $FFFF\ FFFF_H$**

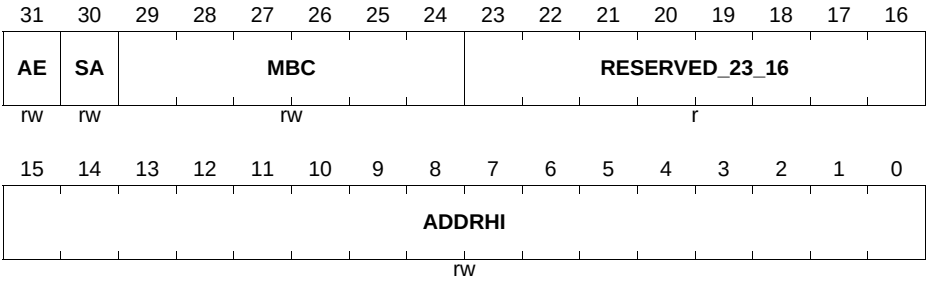


Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address11 [31:0]** This field contains the lower 32 bits of the 12th 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address12_High

The MAC Address12 High register holds the upper 16 bits of the 13th 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address13 Low Register are written. For proper synchronization updates, the consecutive writes to this MAC Address12 Low Register should be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS12_HIGH
Register 40 - MAC Address12 High Register ($10A0_H$) **Reset Value: $0000\ FFFF_H$**



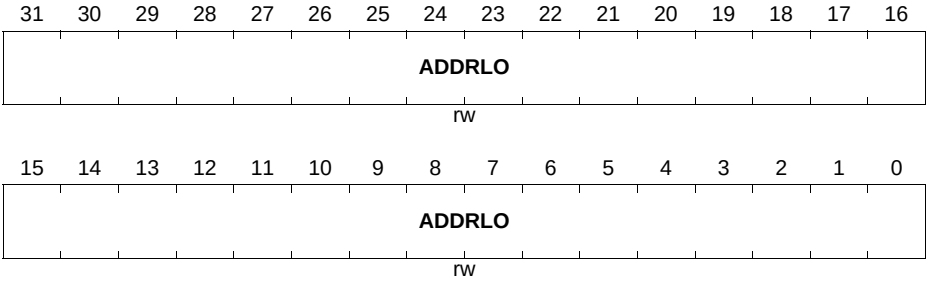
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address12 [47:32]** This field contains the upper 16 bits (47:32) of the 13th 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address12 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address12[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address12[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the 13th MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address12_Low

The MAC Address12 Low register holds the lower 32 bits of the 13th 6-byte MAC address of the station.

ETH_MAC_ADDRESS12_LOW
Register 41 - MAC Address12 Low Register ($10A4_H$) **Reset Value: $FFFF\ FFFF_H$**



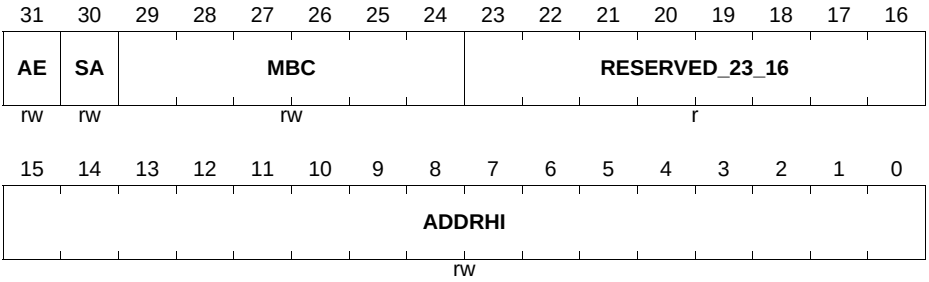
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address12 [31:0]** This field contains the lower 32 bits of the 13th 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address13_High

The MAC Address13 High register holds the upper 16 bits of the 14th 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address13 Low Register are written. For proper synchronization updates, the consecutive writes to this MAC Address13 Low Register should be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS13_HIGH

Register 42 - MAC Address13 High Register ($10A8_H$) **Reset Value: $0000\ FFFF_H$**



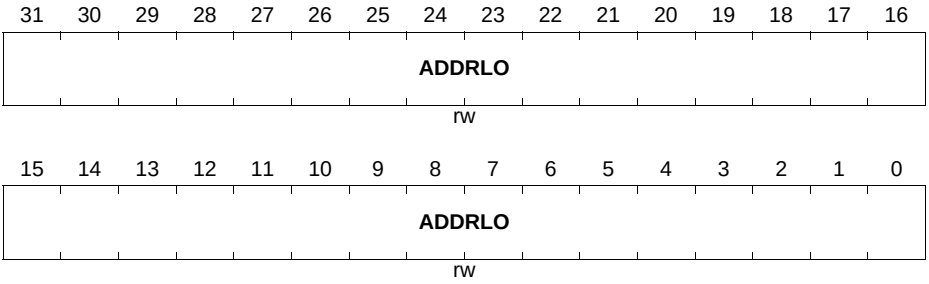
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address13 [47:32]** This field contains the upper 16 bits (47:32) of the 14th 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address13 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address13[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address13[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the 14th MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address13_Low

The MAC Address13 Low register holds the lower 32 bits of the 14th 6-byte MAC address of the station.

ETH_MAC_ADDRESS13_LOW
Register 43 - MAC Address13 Low Register ($10AC_H$) **Reset Value: $FFFF\ FFFF_H$**



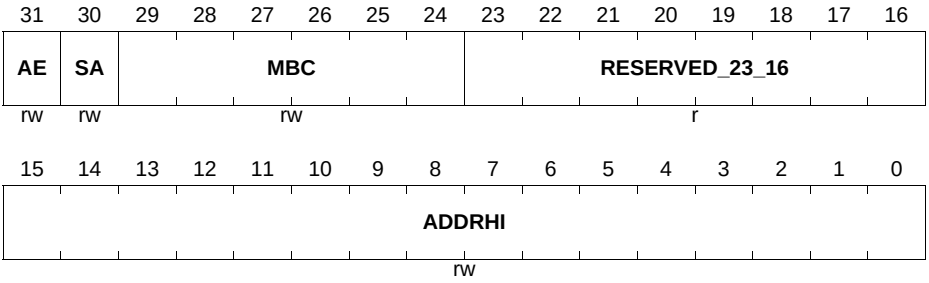
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address13 [31:0]** This field contains the lower 32 bits of the 14th 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address14_High

The MAC Address14 High register holds the upper 16 bits of the 15th 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address15 Low Register are written. For proper synchronization updates, the consecutive writes to this MAC Address14 Low Register should be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS14_HIGH

Register 44 - MAC Address14 High Register ($10B0_H$) **Reset Value: $0000\ FFFF_H$**



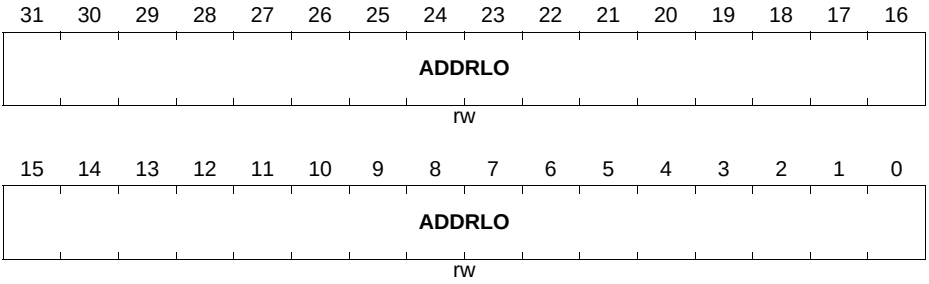
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address14 [47:32]** This field contains the upper 16 bits (47:32) of the 15th 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address14 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address14[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address14[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the 15th MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address14_Low

The MAC Address14 Low register holds the lower 32 bits of the 15th 6-byte MAC address of the station.

ETH_MAC_ADDRESS14_LOW
Register 45 - MAC Address14 Low Register ($10B4_H$) **Reset Value: $FFFF\ FFFF_H$**



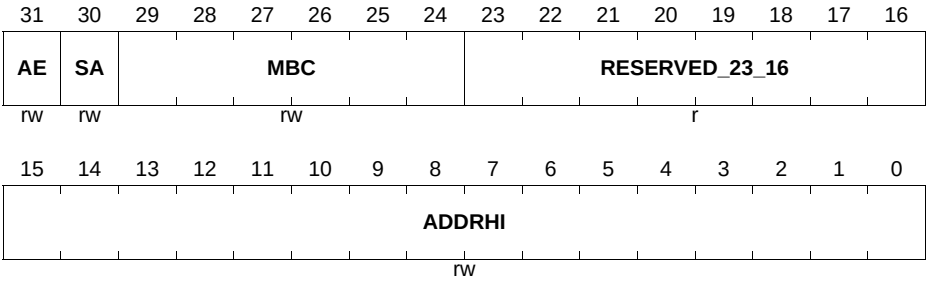
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address14 [31:0]** This field contains the lower 32 bits of the 15th 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address15_High

The MAC Address15 High register holds the upper 16 bits of the 16th 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address15 Low Register are written. For proper synchronization updates, the consecutive writes to this MAC Address15 Low Register should be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS15_HIGH

Register 46 - MAC Address15 High Register ($10B8_H$) **Reset Value: $0000\ FFFF_H$**



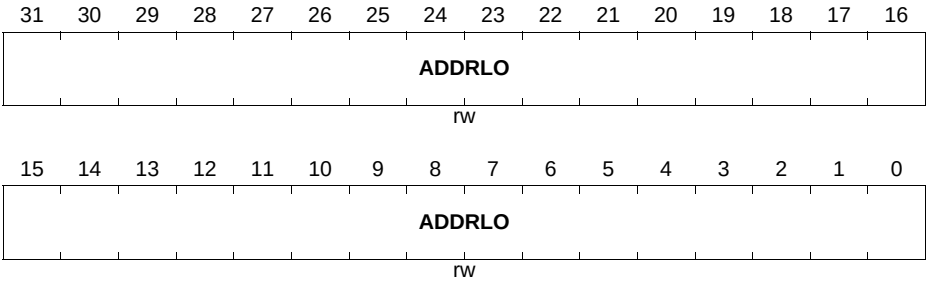
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**ADDRHI** MAC Address15 [47:32] This field contains the upper 16 bits (47:32) of the 16th 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address15 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address15[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address15[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the 16th MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address15_Low

The MAC Address15 Low register holds the lower 32 bits of the 16th 6-byte MAC address of the station.

ETH_MAC_ADDRESS15_LOW
Register 47 - MAC Address15 Low Register ($10BC_H$) **Reset Value: $FFFF\ FFFF_H$**



Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address15 [31:0]** This field contains the lower 32 bits of the 16th 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MMC_Control

The MMC Control register establishes the operating mode of the management counters. Note: The bit 0 (Counters Reset) has higher priority than bit 4 (Counter Preset). Therefore, when the Software tries to set both bits in the same write cycle, all counters are cleared and the bit 4 is not set.

ETH_MMC_CONTROL
Register 64 - MMC Control Register (1100_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED_31_9															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED_31_9							UCDBC	RESERVED_7_6		CNTPRSTLVL	CNTPRST	CNTFREEZ	RSTONRD	CNTSTOPRO	CNTRST
r							rw	r		rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CNTRST	0	rw	**Counters Reset** When this bit is set, all counters are reset. This bit is cleared automatically after one clock cycle.
CNTSTOPRO	1	rw	**Counters Stop Rollover** When this bit is set, after reaching maximum value, the counter does not roll over to zero.
RSTONRD	2	rw	**Reset on Read** When this bit is set, the MMC counters are reset to zero after Read (self-clearing after reset). The counters are cleared when the least significant byte lane (bits[7:0]) is read.
CNTFREEZ	3	rw	**MMC Counter Freeze** When this bit is set, it freezes all MMC counters to their current value. Until this bit is reset to 0, no MMC counter is updated because of any transmitted or received frame. If any MMC counter is read with the Reset on Read bit set, then that counter is also cleared in this mode.

Field	Bits	Type	Description
CNTPRST	4	rw	**Counters Preset** When this bit is set, all counters are initialized or preset to almost full or almost half according to bit 5. This bit is cleared automatically after 1 clock cycle. This bit, along with bit 5, is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full.
CNTPRSTLVL	5	rw	**Full-Half Preset** When low and bit 4 is set, all MMC counters get preset to almost-half value. All octet counters get preset to 0x7FFF_F800 (half - 2KBytes) and all frame-counters gets preset to 0x7FFF_FFF0 (half - 16). When this bit is high and bit 4 is set, all MMC counters get preset to almost-full value. All octet counters get preset to 0xFFFF_F800 (full - 2KBytes) and all frame-counters gets preset to 0xFFFF_FFF0 (full - 16). For 16-bit counters, the almost-half preset values are 0x7800 and 0x7FF0 for the respective octet and frame counters. Similarly, the almost-full preset values for the 16-bit counters are 0xF800 and 0xFFF0.
RESERVED_7_6	[7:6]	r	**RESERVED_7_6**
UCDBC	8	rw	**Update MMC Counters for Dropped Broadcast Frames** When set, this bit enables MAC to update all the related MMC Counters for Broadcast frames dropped due to setting of DBF bit (Disable Broadcast Frames) of MAC Filter Register at offset 0x0004. When reset, MMC Counters are not updated for dropped Broadcast frames.
RESERVED_31_9	[31:9]	r	**RESERVED_31_9**

32-bit Register - MMC_Receive_Interrupt

The MMC Receive Interrupt register maintains the interrupts that are generated when the following happens: * Receive statistic counters reach half of their maximum values (0x8000_0000 for 32-bit counter and 0x8000 for 16-bit counter). * Receive statistic counters cross their maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When the Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Receive Interrupt register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read in order to clear the interrupt bit.

ETH_MMC_RECEIVE_INTERRUPT

Register 65 - MMC Receive Interrupt Register (1104_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED_31_26						RXCTRLFIS	RXRCVERRFIS	RXWDOGFIS	RXVLANGBFIS	RXFOVFIS	RXPAUSFIS	RXORANGEFIS	RXLENERFIS	RXUCGFIS	RX1024TMAXOCTGBFI
r						r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX512T1023OCTGBFIS	RX256T511OCTGBFIS	RX128T255OCTGBFIS	RX65T127OCTGBFIS	RX64OCTGBFIS	RXOSIZEGFIS	RXUSIZEGFIS	RXJABERFIS	RXRUNTFIS	RXALGNERFIS	RXCRCERFIS	RXMCGFIS	RXBCGFIS	RXGOCTIS	RXGBOCTIS	RXGBFRMIS
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
RXGBFRMIS	0	r	**MMC Receive Good Bad Frame Counter Interrupt Status** This bit is set when the rxframecount_bg counter reaches half of the maximum value or the maximum value.
RXGBOCTIS	1	r	**MMC Receive Good Bad Octet Counter Interrupt Status** This bit is set when the rxoctetcount_bg counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RXGOCTIS	2	r	**MMC Receive Good Octet Counter Interrupt Status.** This bit is set when the rxoctetcount_g counter reaches half of the maximum value or the maximum value.
RXBCGFIS	3	r	**MMC Receive Broadcast Good Frame Counter Interrupt Status.** This bit is set when the rxbroadcastframes_g counter reaches half of the maximum value or the maximum value.
RXMCGFIS	4	r	**MMC Receive Multicast Good Frame Counter Interrupt Status** This bit is set when the rxmulticastframes_g counter reaches half of the maximum value or the maximum value.
RXCRCERFIS	5	r	**MMC Receive CRC Error Frame Counter Interrupt Status** This bit is set when the rxcrcerror counter reaches half of the maximum value or the maximum value.
RXALGNERFIS	6	r	**MMC Receive Alignment Error Frame Counter Interrupt Status** This bit is set when the rxalignmenterror counter reaches half of the maximum value or the maximum value.
RXRUNTFIS	7	r	**MMC Receive Runt Frame Counter Interrupt Status** This bit is set when the rxrunterror counter reaches half of the maximum value or the maximum value.
RXJABERFIS	8	r	**MMC Receive Jabber Error Frame Counter Interrupt Status** This bit is set when the rxjabbererror counter reaches half of the maximum value or the maximum value.
RXUSIZEGFIS	9	r	**MMC Receive Undersize Good Frame Counter Interrupt Status** This bit is set when the rxundersize_g counter reaches half of the maximum value or the maximum value.
RXOSIZEGFIS	10	r	**MMC Receive Oversize Good Frame Counter Interrupt Status** This bit is set when the rxoversize_g counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RX64OCTGBFIS	11	r	**MMC Receive 64 Octet Good Bad Frame Counter Interrupt Status** This bit is set when the rx64octets_gb counter reaches half of the maximum value or the maximum value.
RX65T127OCTGBFIS	12	r	**MMC Receive 65 to 127 Octet Good Bad Frame Counter Interrupt Status** This is set when the rx65to127octets_gb counter reaches half of the maximum value or the maximum value.
RX128T255OCTGBFIS	13	r	**MMC Receive 128 to 255 Octet Good Bad Frame Counter Interrupt Status** This bit is set when the rx128to255octets_gb counter reaches half of the maximum value or the maximum value.
RX256T511OCTGBFIS	14	r	**MMC Receive 256 to 511 Octet Good Bad Frame Counter Interrupt Status** This bit is set when the rx256to511octets_gb counter reaches half of the maximum value or the maximum value.
RX512T1023OCTGBFIS	15	r	**MMC Receive 512 to 1023 Octet Good Bad Frame Counter Interrupt Status** This bit is set when the rx512to1023octets_gb counter reaches half of the maximum value or the maximum value.
RX1024TMAXOCTGBFIS	16	r	**MMC Receive 1024 to Maximum Octet Good Bad Frame Counter Interrupt Status** This bit is set when the rx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value.
RXUCGFIS	17	r	**MMC Receive Unicast Good Frame Counter Interrupt Status** This bit is set when the rxunicastframes_gb counter reaches half of the maximum value or the maximum value.
RXLENERFIS	18	r	**MMC Receive Length Error Frame Counter Interrupt Status** This bit is set when the rxlengtherror counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RXORANGEFIS	19	r	**MMC Receive Out Of Range Error Frame Counter Interrupt Status** This bit is set when the rxoutofrangetype counter reaches half of the maximum value or the maximum value.
RXPAUSFIS	20	r	**MMC Receive Pause Frame Counter Interrupt Status** This bit is set when the rxpauseframe counter reaches half of the maximum value or the maximum value.
RXFOVFIS	21	r	**MMC Receive FIFO Overflow Frame Counter Interrupt Status** This bit is set when the rxfifooverflow counter reaches half of the maximum value or the maximum value.
RXVLANGBFIS	22	r	**MMC Receive VLAN Good Bad Frame Counter Interrupt Status** This bit is set when the rxvlanframes_gb counter reaches half of the maximum value or the maximum value.
RXWDOGFIS	23	r	**MMC Receive Watchdog Error Frame Counter Interrupt Status** This bit is set when the rxwatchdogerror counter reaches half of the maximum value or the maximum value.
RXRCVERRFIS	24	r	**MMC Receive Error Frame Counter Interrupt Status** This bit is set when the rxrcverror counter reaches half of the maximum value or the maximum value.
RXCTRLFIS	25	r	**MMC Receive Control Frame Counter Interrupt Status** This bit is set when the rxctrlframes_g counter reaches half of the maximum value or the maximum value.
RESERVED_31_26	[31:26]	r	**RESERVED_31_26**

32-bit Register - MMC_Transmit_Interrupt

The MMC Transmit Interrupt register maintains the interrupts generated when transmit statistic counters reach half of their maximum values (0x8000_0000 for 32-bit counter and 0x8000 for 16-bit counter), and the maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Transmit Interrupt register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read in order to clear the interrupt bit.

ETH_MMC_TRANSMIT_INTERRUPT

Register 66 - MMC Transmit Interrupt Register (1108_H) Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED_31_26						TXOSIZEGFIS	TXVLANGFIS	TXPAUSFIS	TXEXDEFFIS	TXGFRMIS	TXGOCTIS	TXCARERFIS	TXEXCOLFIS	TXLATCOLFIS	TXDEFFIS
r						r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMCOLGFIS	TXSCOLGFIS	TXUFLOWERFIS	TXBCGBFIS	TXMCGBFIS	TXUCGBFIS	TX1024TMAXOCTGBFI	TX512T1023OCTGBFIS	TX256T511OCTGBFIS	TX128T255OCTGBFIS	TX65T127OCTGBFIS	TX64OCTGBFIS	TXMCGFIS	TXBCGFIS	TXGBFRMIS	TXGBOCTIS
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
TXGBOCTIS	0	r	**MMC Transmit Good Bad Octet Counter Interrupt Status** This bit is set when the txoctetcount_gb counter reaches half of the maximum value or the maximum value.
TXGBFRMIS	1	r	**MMC Transmit Good Bad Frame Counter Interrupt Status** This bit is set when the txframecount_gb counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
TXBCGFIS	2	r	**MMC Transmit Broadcast Good Frame Counter Interrupt Status** This bit is set when the txbroadcastframes_g counter reaches half of the maximum value or the maximum value.
TXMCGFIS	3	r	**MMC Transmit Multicast Good Frame Counter Interrupt Status** This bit is set when the txmulticastframes_g counter reaches half of the maximum value or the maximum value.
TX64OCTGBFIS	4	r	**MMC Transmit 64 Octet Good Bad Frame Counter Interrupt Status.** This bit is set when the tx64octets_gb counter reaches half of the maximum value or the maximum value.
TX65T127OCTGBFIS	5	r	**MMC Transmit 65 to 127 Octet Good Bad Frame Counter Interrupt Status** This bit is set when the tx65to127octets_gb counter reaches half of the maximum value or the maximum value.
TX128T255OCTGBFIS	6	r	**MMC Transmit 128 to 255 Octet Good Bad Frame Counter Interrupt Status** This bit is set when the tx128to255octets_gb counter reaches half of the maximum value or the maximum value.
TX256T511OCTGBFIS	7	r	**MMC Transmit 256 to 511 Octet Good Bad Frame Counter Interrupt Status** This bit is set when the tx256to511octets_gb counter reaches half of the maximum value or the maximum value.
TX512T1023OCTGBFIS	8	r	**MMC Transmit 512 to 1023 Octet Good Bad Frame Counter Interrupt Status** This bit is set when the tx512to1023octets_gb counter reaches half of the maximum value or the maximum value.
TX1024TMAXOCTGBFIS	9	r	**MMC Transmit 1024 to Maximum Octet Good Bad Frame Counter Interrupt Status** This bit is set when the tx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
TXUCGBFIS	10	r	**MMC Transmit Unicast Good Bad Frame Counter Interrupt Status** This bit is set when the txunicastframes_gb counter reaches half of the maximum value or the maximum value.
TXMCGBFIS	11	r	**MMC Transmit Multicast Good Bad Frame Counter Interrupt Status** This bit is set when the txmulticastframes_gb counter reaches half of the maximum value or the maximum value.
TXBCGBFIS	12	r	**MMC Transmit Broadcast Good Bad Frame Counter Interrupt Status** This bit is set when the txbroadcastframes_gb counter reaches half of the maximum value or the maximum value.
TXUFLOWERFIS	13	r	**MMC Transmit Underflow Error Frame Counter Interrupt Status** This bit is set when the txunderflowerror counter reaches half of the maximum value or the maximum value.
TXSCOLGFIS	14	r	**MMC Transmit Single Collision Good Frame Counter Interrupt Status** This bit is set when the txsinglecol_g counter reaches half of the maximum value or the maximum value.
TXMCOLGFIS	15	r	**MMC Transmit Multiple Collision Good Frame Counter Interrupt Status** This bit is set when the txmulticol_g counter reaches half of the maximum value or the maximum value.
TXDEFFIS	16	r	**MMC Transmit Deferred Frame Counter Interrupt Status** This bit is set when the txdeferred counter reaches half of the maximum value or the maximum value.
TXLATCOLFIS	17	r	**MMC Transmit Late Collision Frame Counter Interrupt Status** This bit is set when the txlatecol counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
TXEXCOLFIS	18	r	**MMC Transmit Excessive Collision Frame Counter Interrupt Status** This bit is set when the txexcesscol counter reaches half of the maximum value or the maximum value.
TXCARERFIS	19	r	**MMC Transmit Carrier Error Frame Counter Interrupt Status** This bit is set when the txcarriererror counter reaches half of the maximum value or the maximum value.
TXGOCTIS	20	r	**MMC Transmit Good Octet Counter Interrupt Status** This bit is set when the txoctetcount_g counter reaches half of the maximum value or the maximum value.
TXGFRMIS	21	r	**MMC Transmit Good Frame Counter Interrupt Status** This bit is set when the txframecount_g counter reaches half of the maximum value or the maximum value.
TXEXDEFFIS	22	r	**MMC Transmit Excessive Deferral Frame Counter Interrupt Status** This bit is set when the txexcessdef counter reaches half of the maximum value or the maximum value.
TXPAUSFIS	23	r	**MMC Transmit Pause Frame Counter Interrupt Status** This bit is set when the txpauseframeserror counter reaches half of the maximum value or the maximum value.
TXVLANGFIS	24	r	**MMC Transmit VLAN Good Frame Counter Interrupt Status** This bit is set when the txvlanframes_g counter reaches half of the maximum value or the maximum value.
TXOSIZEGFIS	25	r	**MMC Transmit Oversize Good Frame Counter Interrupt Status** This bit is set when the txoversize_g counter reaches half of the maximum value or the maximum value.
RESERVED_31_26	[31:26]	r	**RESERVED_31_26**

32-bit Register - MMC_Receive_Interrupt_Mask

ETH_MMC_RECEIVE_INTERRUPT_MASK

- **($110C_H$)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED_31_26						RXCTRLFIM	RXRCVERRFIM	RXWDOGFIM	RXVLANGBFIM	RXFOVFIM	RXPAUSFIM	RXORANGEFIM	RXLENERFIM	RXUCGFIM	RX1024TMAXOCTGBFI
r						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX512T1023OCTGBFIM	RX256T511OCTGBFIM	RX128T255OCTGBFIM	RX65T127OCTGBFIM	RX64OCTGBFIM	RXOSIZEGFIM	RXUSIZEGFIM	RXJABERFIM	RXRUNTFIM	RXALGNERFIM	RXCRCERFIM	RXMCGFIM	RXBCGFIM	RXGOCTIM	RXGBOCTIM	RXGBFRMIM
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
RXGBFRMIM	0	rw	**MMC Receive Good Bad Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxframecount_gb counter reaches half of the maximum value or the maximum value.
RXGBOCTIM	1	rw	**MMC Receive Good Bad Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxoctetcount_gb counter reaches half of the maximum value or the maximum value.
RXGOCTIM	2	rw	**MMC Receive Good Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxoctetcount_g counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RXBCGFIM	3	rw	**MMC Receive Broadcast Good Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxbroadcastframes_g counter reaches half of the maximum value or the maximum value.
RXMCGFIM	4	rw	**MMC Receive Multicast Good Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxmulticastframes_g counter reaches half of the maximum value or the maximum value.
RXCRCERFIM	5	rw	**MMC Receive CRC Error Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxcrcerror counter reaches half of the maximum value or the maximum value.
RXALGNERFIM	6	rw	**MMC Receive Alignment Error Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxalignmenterror counter reaches half of the maximum value or the maximum value.
RXRUNTFIM	7	rw	**MMC Receive Runt Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxrunterror counter reaches half of the maximum value or the maximum value.
RXJABERFIM	8	rw	**MMC Receive Jabber Error Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxjabbererror counter reaches half of the maximum value or the maximum value.
RXUSIZEGFIM	9	rw	**MMC Receive Undersize Good Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxundersize_g counter reaches half of the maximum value or the maximum value.
RXOSIZEGFIM	10	rw	**MMC Receive Oversize Good Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxoversize_g counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RX64OCTGBFIM	11	rw	**MMC Receive 64 Octet Good Bad Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rx64octets_gb counter reaches half of the maximum value or the maximum value.
RX65T127OCTGBFIM	12	rw	**MMC Receive 65 to 127 Octet Good Bad Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rx65to127octets_gb counter reaches half of the maximum value or the maximum value.
RX128T255OCTGBFIM	13	rw	**MMC Receive 128 to 255 Octet Good Bad Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rx128to255octets_gb counter reaches half of the maximum value or the maximum value.
RX256T511OCTGBFIM	14	rw	**MMC Receive 256 to 511 Octet Good Bad Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rx256to511octets_gb counter reaches half of the maximum value or the maximum value.
RX512T1023OCTGBFIM	15	rw	**MMC Receive 512 to 1023 Octet Good Bad Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rx512to1023octets_gb counter reaches half of the maximum value or the maximum value.
RX1024TMAXOCTGBFIM	16	rw	**MMC Receive 1024 to Maximum Octet Good Bad Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value.
RXUCGFIM	17	rw	**MMC Receive Unicast Good Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxunicastframes_g counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RXLENERFIM	18	rw	**MMC Receive Length Error Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxlengtherror counter reaches half of the maximum value or the maximum value.
RXORANGEFIM	19	rw	**MMC Receive Out Of Range Error Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxoutofrangetype counter reaches half of the maximum value or the maximum value.
RXPAUSFIM	20	rw	**MMC Receive Pause Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxpauseframes counter reaches half of the maximum value or the maximum value.
RXFOVFIM	21	rw	**MMC Receive FIFO Overflow Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxfifooverflow counter reaches half of the maximum value or the maximum value.
RXVLANGBFIM	22	rw	**MMC Receive VLAN Good Bad Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxvlanframes_gb counter reaches half of the maximum value or the maximum value.
RXWDOGFIM	23	rw	**MMC Receive Watchdog Error Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxwatchdog counter reaches half of the maximum value or the maximum value.
RXRCVERRFIM	24	rw	**MMC Receive Error Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxrcverror error counter reaches half the maximum value, and also when it reaches the maximum value.
RXCTRLFIM	25	rw	**MMC Receive Control Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxctrlframes counter reaches half the maximum value, and also when it reaches the maximum value.

Field	Bits	Type	Description
RESERVED_31_26	[31:26]	r	**RESERVED_31_26**

32-bit Register - MMC_Transmit_Interrupt_Mask

The MMC Transmit Interrupt Mask register maintains the masks for the interrupts generated when the transmit statistic counters reach half of their maximum value or maximum value. This register is 32-bits wide.

ETH_MMC_TRANSMIT_INTERRUPT_MASK
Register 68 - MMC Transmit Interrupt Mask Register (1110_H) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED_31_26						TXOSIZEGFIM	TXVLANGFIM	TXPAUSFIM	TXEXDEFFIM	TXGFRMIM	TXGOCTIM	TXCARERFIM	TXEXCOLFIM	TXLATCOLFIM	TXDEFFIM
r						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMCOLGFIM	TXSCOLGFIM	TXUFLOWERFIM	TXBCGBFIM	TXMCGBFIM	TXUCGBFIM	TX1024TMAXOCTGBFI	TX512T1023OCTGBFIM	TX256T511OCTGBFIM	TX128T255OCTGBFIM	TX65T127OCTGBFIM	TX64OCTGBFIM	TXMCGFIM	TXBCGFIM	TXGBFRMIM	TXGBOCTIM
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
TXGBOCTIM	0	rw	**MMC Transmit Good Bad Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the txoctetcount_gb counter reaches half of the maximum value or the maximum value.
TXGBFRMIM	1	rw	**MMC Transmit Good Bad Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the txframecount_gb counter reaches half of the maximum value or the maximum value.
TXBCGFIM	2	rw	**MMC Transmit Broadcast Good Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the txbroadcastframes_g counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
TXMCGFIM	3	rw	**MMC Transmit Multicast Good Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the txmulticastframes_g counter reaches half of the maximum value or the maximum value.
TX64OCTGBFIM	4	rw	**MMC Transmit 64 Octet Good Bad Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the tx64octets_gb counter reaches half of the maximum value or the maximum value.
TX65T127OCTGBFIM	5	rw	**MMC Transmit 65 to 127 Octet Good Bad Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the tx65to127octets_gb counter reaches half of the maximum value or the maximum value.
TX128T255OCTGBFIM	6	rw	**MMC Transmit 128 to 255 Octet Good Bad Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the tx128to255octets_gb counter reaches half of the maximum value or the maximum value.
TX256T511OCTGBFIM	7	rw	**MMC Transmit 256 to 511 Octet Good Bad Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the tx256to511octets_gb counter reaches half of the maximum value or the maximum value.
TX512T1023OCTGBFIM	8	rw	**MMC Transmit 512 to 1023 Octet Good Bad Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the tx512to1023octets_gb counter reaches half of the maximum value or the maximum value.
TX1024TMAXOCTGBFIM	9	rw	**MMC Transmit 1024 to Maximum Octet Good Bad Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the tx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
TXUCGBFIM	10	rw	**MMC Transmit Unicast Good Bad Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the txunicastframes_gb counter reaches half of the maximum value or the maximum value.
TXMCGBFIM	11	rw	**MMC Transmit Multicast Good Bad Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the txmulticastframes_gb counter reaches half of the maximum value or the maximum value.
TXBCGBFIM	12	rw	**MMC Transmit Broadcast Good Bad Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the txbroadcastframes_gb counter reaches half of the maximum value or the maximum value.
TXUFLOWERFIM	13	rw	**MMC Transmit Underflow Error Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the txunderflowerror counter reaches half of the maximum value or the maximum value.
TXSCOLGFIM	14	rw	**MMC Transmit Single Collision Good Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the txsinglecol_g counter reaches half of the maximum value or the maximum value.
TXMCOLGFIM	15	rw	**MMC Transmit Multiple Collision Good Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the txmulticol_g counter reaches half of the maximum value or the maximum value.
TXDEFFIM	16	rw	**MMC Transmit Deferred Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the txdeferred counter reaches half of the maximum value or the maximum value.

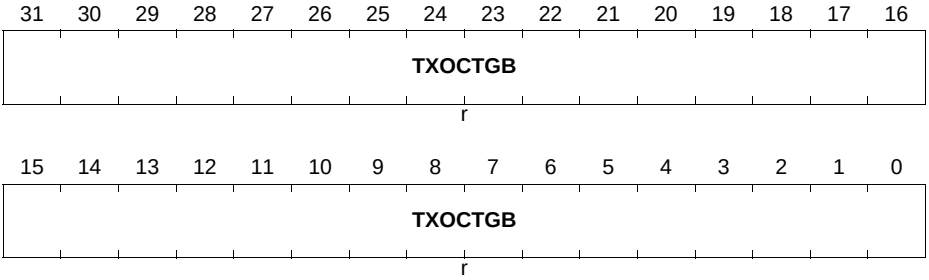
Field	Bits	Type	Description
TXLATCOLFIM	17	rw	**MMC Transmit Late Collision Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the txlatecol counter reaches half of the maximum value or the maximum value.
TXEXCOLFIM	18	rw	**MMC Transmit Excessive Collision Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the txexcesscol counter reaches half of the maximum value or the maximum value.
TXCARERFIM	19	rw	**MMC Transmit Carrier Error Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the txcarriererror counter reaches half of the maximum value or the maximum value.
TXGOCTIM	20	rw	**MMC Transmit Good Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the txoctetcount_g counter reaches half of the maximum value or the maximum value.
TXGFRMIM	21	rw	**MMC Transmit Good Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the txframecount_g counter reaches half of the maximum value or the maximum value.
TXEXDEFFIM	22	rw	**MMC Transmit Excessive Deferral Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the txexcessdef counter reaches half of the maximum value or the maximum value.
TXPAUSFIM	23	rw	**MMC Transmit Pause Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the txpauseframes counter reaches half of the maximum value or the maximum value.
TXVLANGFIM	24	rw	**MMC Transmit VLAN Good Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the txvlanframes_g counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
TXOSIZEGFIM	25	rw	**MMC Transmit Oversize Good Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the txoversize_g counter reaches half of the maximum value or the maximum value.
RESERVED_31_26	[31:26]	r	**RESERVED_31_26**

32-bit Register - Tx_Octet_Count_Good_Bad

This register maintains the number of bytes transmitted in good and bad frames exclusive of preamble and retried bytes.

ETH_TX_OCTET_COUNT_GOOD_BAD
Register 69 - Transmit Octet Count for Good and Bad Frames (1114_H) Reset Value: $0000\ 0000_H$

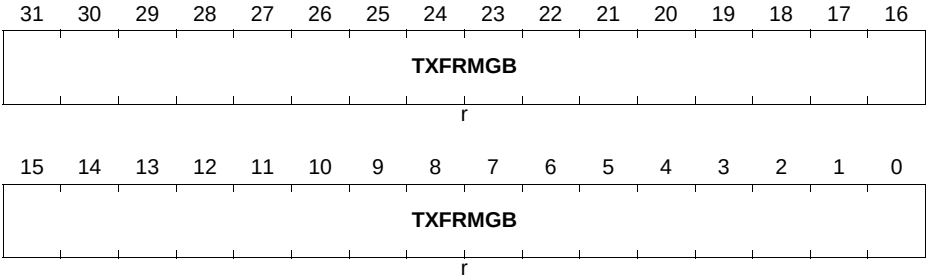


Field	Bits	Type	Description
TXOCTGB	[31:0]	r	**TXOCTGB** This field indicates the number of bytes transmitted in good and bad frames exclusive of preamble and retried bytes.

32-bit Register - Tx_Frame_Count_Good_Bad

This register maintains the number of good and bad frames transmitted, exclusive of retried frames.

ETH_TX_FRAME_COUNT_GOOD_BAD
Register 70 - Transmit Frame Count for Good and Bad Frames (1118_H) Reset Value: $0000\ 0000_H$

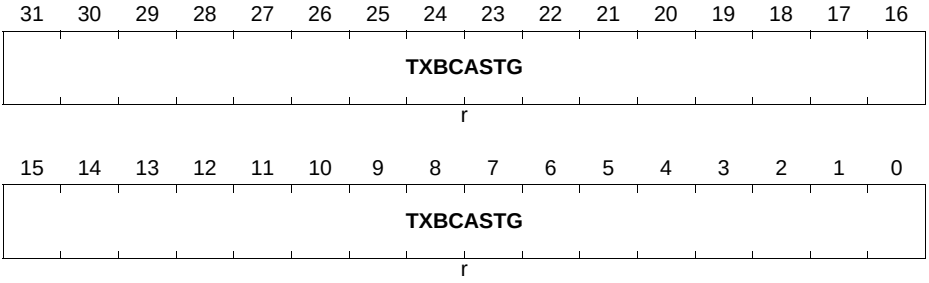


Field	Bits	Type	Description
TXFRMGB	[31:0]	r	**TXFRMGB** This field indicates the number of good and bad frames transmitted, exclusive of retried frames

32-bit Register - Tx_Broadcast_Frames_Good

This register maintains the number of transmitted good broadcast frames.

ETH_TX_BROADCAST_FRAMES_GOOD
Register 71 - Transmit Frame Count for Good Broadcast Frames ($111C_H$) Reset Value: $0000\ 0000_H$

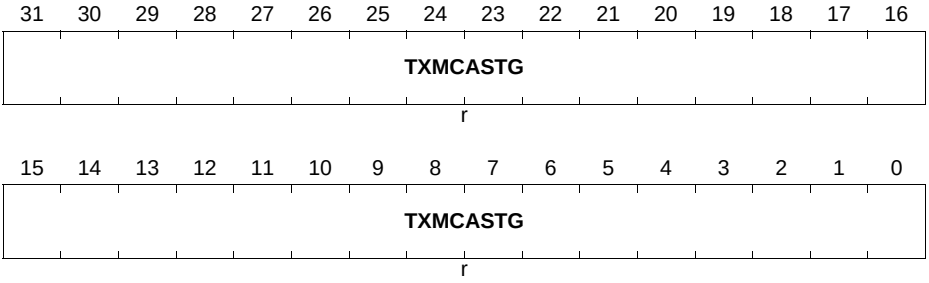


Field	Bits	Type	Description
TXBCASTG	[31:0]	r	**TXBCASTG** This field indicates the number of transmitted good broadcast frames.

32-bit Register - Tx_Multicast_Frames_Good

This register maintains the number of transmitted good multicast frames.

ETH_TX_MULTICAST_FRAMES_GOOD
Register 72 - Transmit Frame Count for Good Multicast Frames (1120_H) Reset Value: $0000\ 0000_H$

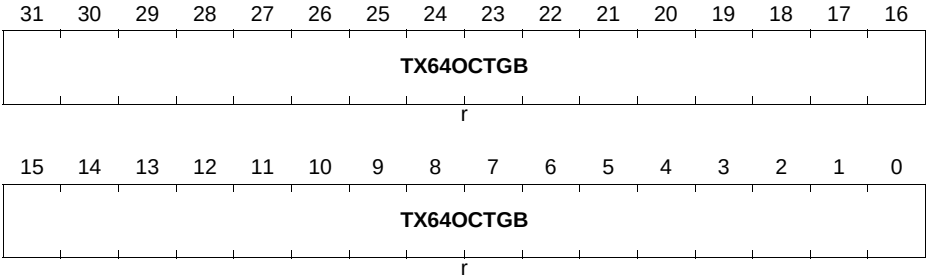


Field	Bits	Type	Description
TXMCASTG	[31:0]	r	**TXMCASTG** This field indicates the number of transmitted good multicast frames.

32-bit Register - Tx_64Octets_Frames_Good_Bad

This register maintains the number of transmitted good and bad frames with length of 64 bytes, exclusive of preamble and retried frames.

ETH_TX_64OCTETS_FRAMES_GOOD_BAD
Register 73 - Transmit Octet Count for Good and Bad 64 Byte Frames (1124_H)
Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
TX64OCTGB	[31:0]	r	**TX64OCTGB** This field indicates the number of transmitted good and bad frames with length of 64 bytes, exclusive of preamble and retried frames.

32-bit Register - Tx_65To127Octets_Frames_Good_Bad

This register maintains the number of transmitted good and bad frames with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried frames.

ETH_TX_65TO127OCTETS_FRAMES_GOOD_BAD
Register 74 - Transmit Octet Count for Good and Bad 65 to 127 Bytes Frames (1128_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
TX65_127 OCTGB	[31:0]	r	**TX65_127OCTGB** This field indicates the number of transmitted good and bad frames with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried frames.

32-bit Register - Tx_128To255Octets_Frames_Good_Bad

This register maintains the number of transmitted good and bad frames with length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried frames.

ETH_TX_128TO255OCTETS_FRAMES_GOOD_BAD
Register 75 - Transmit Octet Count for Good and Bad 128 to 255 Bytes Frames ($112C_H$) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
TX128_255OCTGB	[31:0]	r	**TX128_255OCTGB** This field indicates the number of transmitted good and bad frames with length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried frames.

32-bit Register - Tx_256To511Octets_Frames_Good_Bad

This register maintains the number of transmitted good and bad frames with length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried frames.

ETH_TX_256TO511OCTETS_FRAMES_GOOD_BAD
Register 76 - Transmit Octet Count for Good and Bad 256 to 511 Bytes Frames (1130_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
TX256_511OCTGB	[31:0]	r	**TX256_511OCTGB** This field indicates the number of transmitted good and bad frames with length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried frames.

32-bit Register - Tx_512To1023Octets_Frames_Good_Bad

This register maintains the number of transmitted good and bad frames with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble and retried frames.

ETH_TX_512TO1023OCTETS_FRAMES_GOOD_BAD
Register 77 - Transmit Octet Count for Good and Bad 512 to 1023 Bytes Frames (1134_H) **Reset Value: $0000\ 0000_H$**

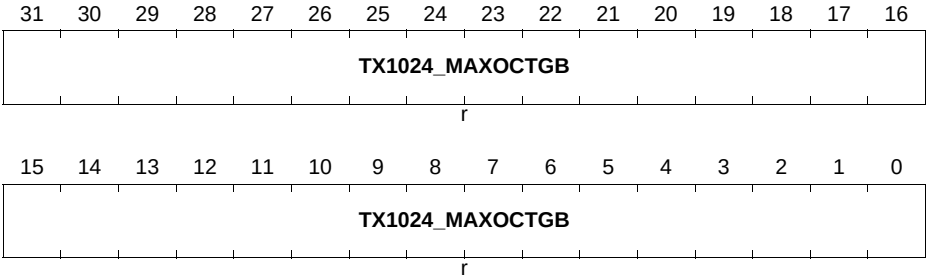


Field	Bits	Type	Description
TX512_1023OCTGB	[31:0]	r	**TX512_1023OCTGB** This field indicates the number of transmitted good and bad frames with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble and retried frames.

32-bit Register - Tx_1024ToMaxOctets_Frames_Good_Bad

This register maintains the number of transmitted good and bad frames with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames.

ETH_TX_1024TOMAXOCTETS_FRAMES_GOOD_BAD
Register 78 - Transmit Octet Count for Good and Bad 1024 to Maxsize Bytes Frames (1138_H) Reset Value: $0000\ 0000_H$

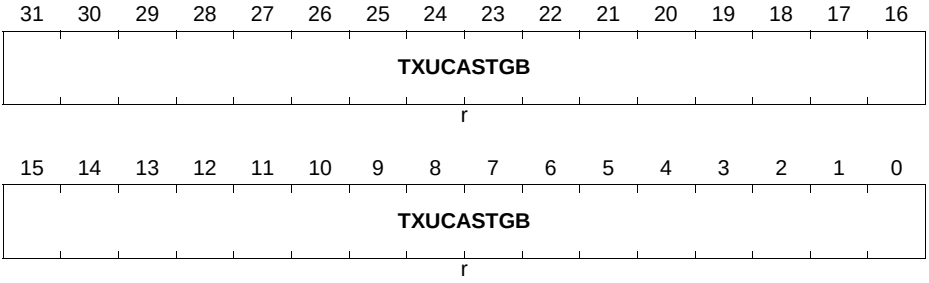


Field	Bits	Type	Description
TX1024_MAXOCTGB	[31:0]	r	**TX1024_MAXOCTGB** This field indicates the number of good and bad frames transmitted with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames.

32-bit Register - Tx_Unicast_Frames_Good_Bad

This register maintains the number of transmitted good and bad unicast frames.

ETH_TX_UNICAST_FRAMES_GOOD_BAD
Register 79 - Transmit Frame Count for Good and Bad Unicast Frames ($113C_H$)
Reset Value: $0000\ 0000_H$

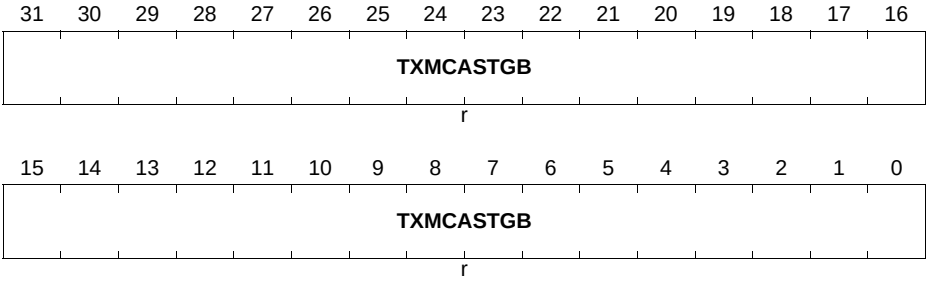


Field	Bits	Type	Description
TXUCASTGB	[31:0]	r	**TXUCASTGB** This field indicates the number of transmitted good and bad unicast frames.

32-bit Register - Tx_Multicast_Frames_Good_Bad

This register maintains the number of transmitted good and bad multicast frames.

ETH_TX_MULTICAST_FRAMES_GOOD_BAD
Register 80 - Transmit Frame Count for Good and Bad Multicast Frames (1140_H)
Reset Value: $0000\ 0000_H$

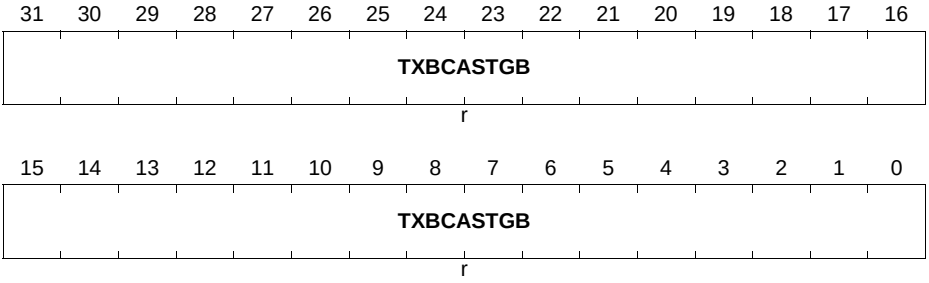


Field	Bits	Type	Description
TXMCASTGB	[31:0]	r	**TXMCASTGB** This field indicates the number of transmitted good and bad multicast frames.

32-bit Register - Tx_Broadcast_Frames_Good_Bad

This register maintains the number of transmitted good and bad broadcast frames.

ETH_TX_BROADCAST_FRAMES_GOOD_BAD
Register 81 - Transmit Frame Count for Good and Bad Broadcast Frames (1144_H)
Reset Value: $0000\ 0000_H$

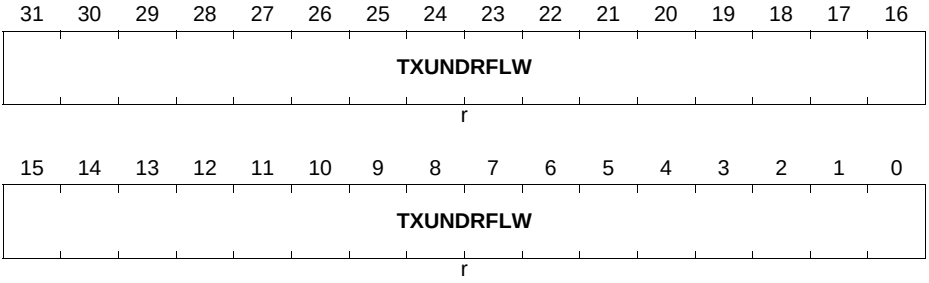


Field	Bits	Type	Description
TXBCASTGB	[31:0]	r	**TXBCASTGB** This field indicates the number of transmitted good and bad broadcast frames.

32-bit Register - Tx_Underflow_Error_Frames

This register maintains the number of frames aborted because of frame underflow error.

ETH_TX_UNDERFLOW_ERROR_FRAMES
Register 82 - Transmit Frame Count for Underflow Error Frames (1148_H) Reset Value: $0000\ 0000_H$

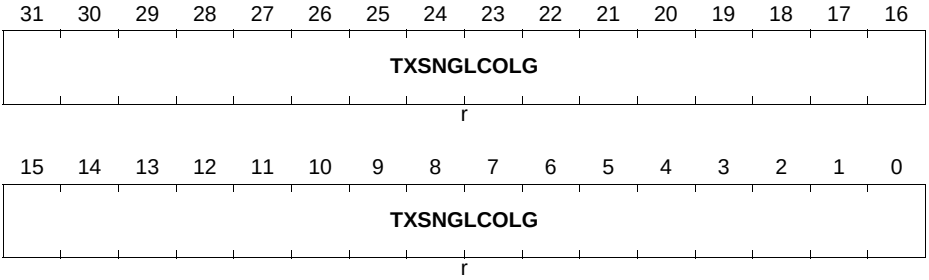


Field	Bits	Type	Description
TXUNDRFLW	[31:0]	r	**TXUNDRFLW** This field indicates the number of frames aborted because of frame underflow error.

32-bit Register - Tx_Single_Collision_Good_Frames

This register maintains the number of successfully transmitted frames after a single collision in the half-duplex mode.

ETH_TX_SINGLE_COLLISION_GOOD_FRAMES
Register 83 - Transmit Frame Count for Frames Transmitted after Single Collision ($114C_H$) **Reset Value: $0000\ 0000_H$**

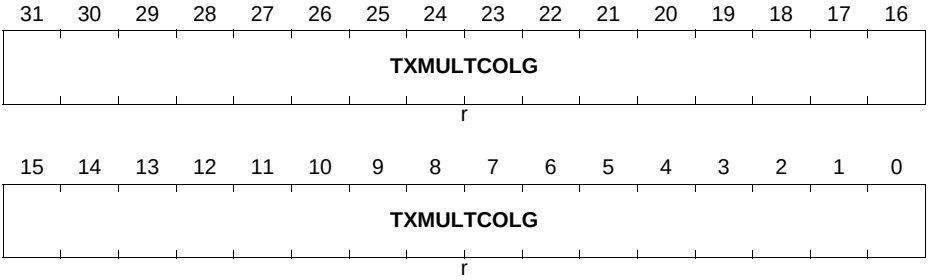


Field	Bits	Type	Description
TXSNGLCOLG	[31:0]	r	**TXSNGLCOLG** This field indicates the number of successfully transmitted frames after a single collision in the half-duplex mode.

32-bit Register - Tx_Multiple_Collision_Good_Frames

This register maintains the number of successfully transmitted frames after multiple collisions in the half-duplex mode.

ETH_TX_MULTIPLE_COLLISION_GOOD_FRAMES
Register 84 - Transmit Frame Count for Frames Transmitted after Multiple Collision (1150_H) Reset Value: $0000\ 0000_H$

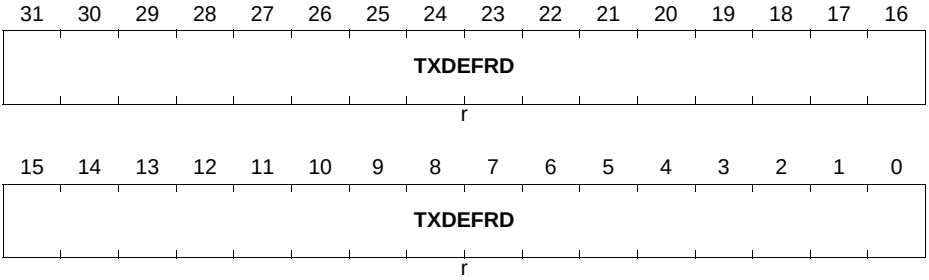


Field	Bits	Type	Description
TXMULTCOLG	[31:0]	r	**TXMULTCOLG** This field indicates the number of successfully transmitted frames after multiple collisions in the half-duplex mode.

32-bit Register - Tx_Deferred_Frames

This register maintains the number of successfully transmitted frames after a deferral in the half-duplex mode.

ETH_TX_DEFERRED_FRAMES
Register 85 - Transmit Frame Count for Deferred Frames (1154_H) Reset Value: $0000\ 0000_H$

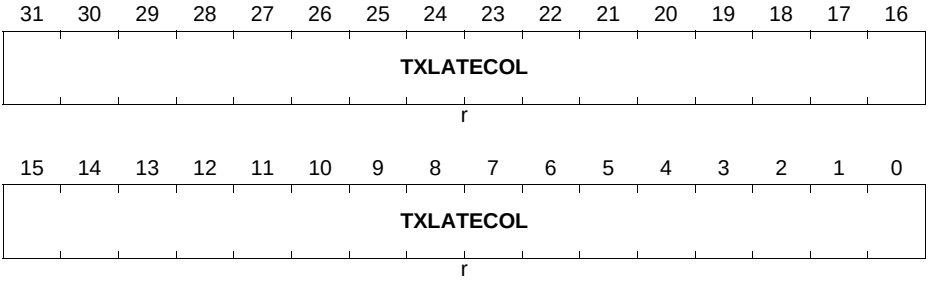


Field	Bits	Type	Description
TXDEFRD	[31:0]	r	**TXDEFRD** This field indicates the number of successfully transmitted frames after a deferral in the half-duplex mode.

32-bit Register - Tx_Late_Collision_Frames

This register maintains the number of frames aborted because of late collision error.

ETH_TX_LATE_COLLISION_FRAMES
Register 86 - Transmit Frame Count for Late Collision Error Frames (1158_H) Reset Value: $0000\ 0000_H$

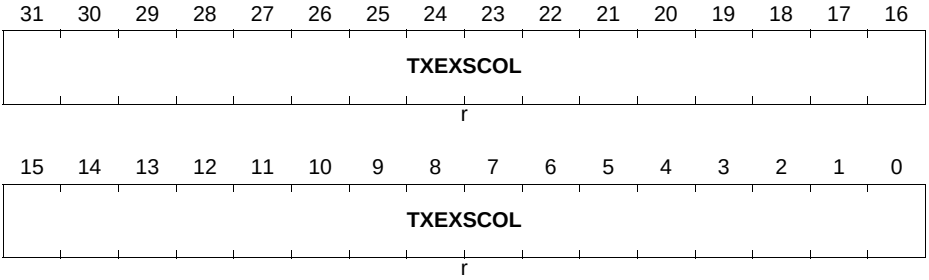


Field	Bits	Type	Description
TXLATECOL	[31:0]	r	**TXLATECOL** This field indicates the number of frames aborted because of late collision error.

32-bit Register - Tx_Excessive_Collision_Frames

This register maintains the number of frames aborted because of excessive (16) collision error.

ETH_TX_EXCESSIVE_COLLISION_FRAMES
Register 87 - Transmit Frame Count for Excessive Collision Error Frames ($115C_H$) **Reset Value: $0000\ 0000_H$**

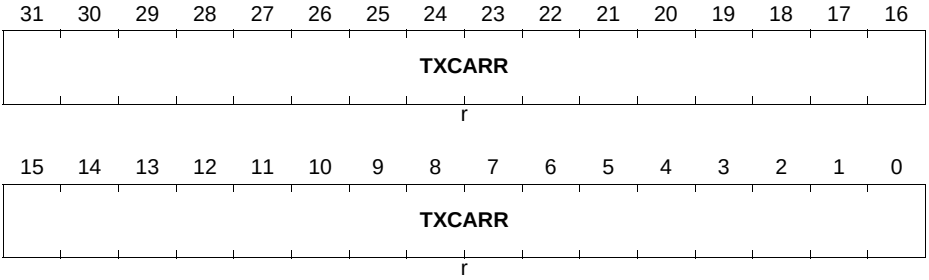


Field	Bits	Type	Description
TXEXSCOL	[31:0]	r	**TXEXSCOL** This field indicates the number of frames aborted because of excessive (16) collision error.

32-bit Register - Tx_Carrier_Error_Frames

This register maintains the number of frames aborted because of carrier sense error (no carrier or loss of carrier).

ETH_TX_CARRIER_ERROR_FRAMES
Register 88 - Transmit Frame Count for Carrier Sense Error Frames (1160_H) Reset Value: $0000\ 0000_H$

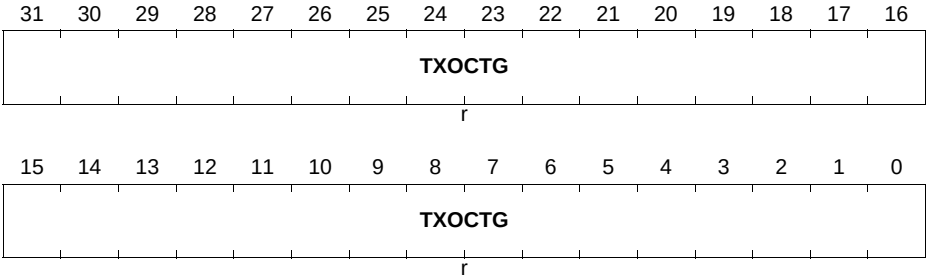


Field	Bits	Type	Description
TXCARR	[31:0]	r	**TXCARR** This field indicates the number of frames aborted because of carrier sense error (no carrier or loss of carrier).

32-bit Register - Tx_Octet_Count_Good

This register maintains the number of bytes transmitted, exclusive of preamble, in good frames.

ETH_TX_OCTET_COUNT_GOOD
Register 89 - Transmit Octet Count for Good Frames (1164_H) Reset Value: $0000\ 0000_H$



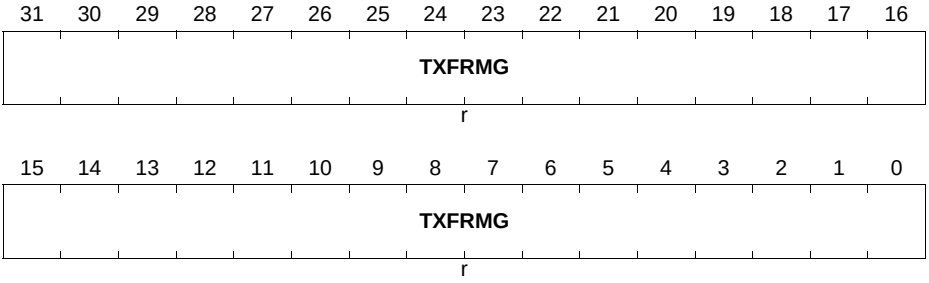
Field	Bits	Type	Description
TXOCTG	[31:0]	r	**TXOCTG** This field indicates the number of bytes transmitted, exclusive of preamble, in good frames.

32-bit Register - Tx_Frame_Count_Good

This register maintains the number of transmitted good frames, exclusive of preamble.

ETH_TX_FRAME_COUNT_GOOD

Register 90 - Transmit Frame Count for Good Frames (1168_H) Reset Value: 0000 0000_H

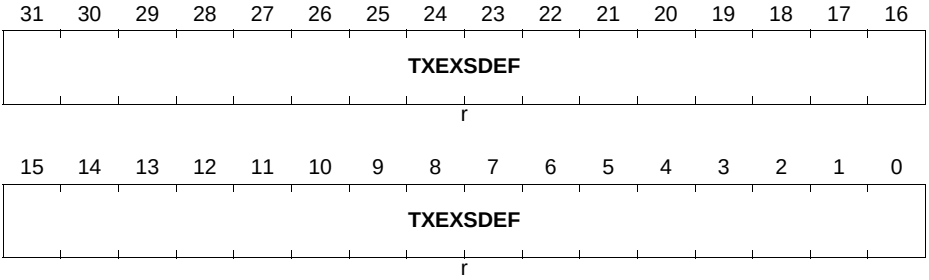


Field	Bits	Type	Description
TXFRMG	[31:0]	r	**TXFRMG** This field indicates the number of transmitted good frames, exclusive of preamble.

32-bit Register - Tx_Excessive_Deferral_Error

This register maintains the number of frames aborted because of excessive deferral error, that is, frames deferred for more than two max-sized frame times.

ETH_TX_EXCESSIVE_DEFERRAL_ERROR
Register 91 - Transmit Frame Count for Excessive Deferral Error Frames ($116C_H$)
Reset Value: $0000\ 0000_H$

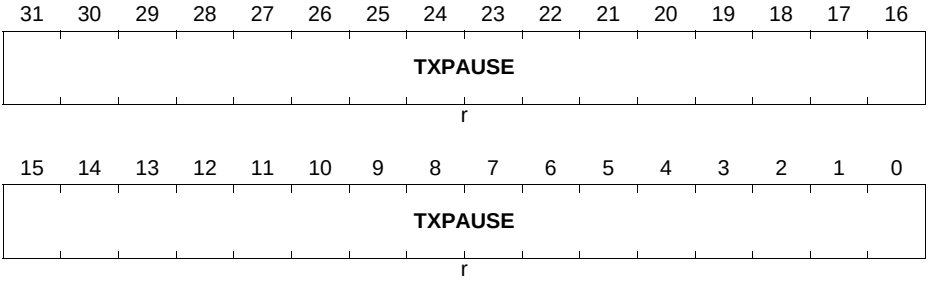


Field	Bits	Type	Description
TXEXSDEF	[31:0]	r	**TXEXSDEF** This field indicates the number of frames aborted because of excessive deferral error, that is, frames deferred for more than two max-sized frame times.

32-bit Register - Tx_Pause_Frames

This register maintains the number of transmitted good PAUSE frames.

ETH_TX_PAUSE_FRAMES
Register 92 - Transmit Frame Count for Good PAUSE Frames (1170_H) Reset Value: $0000\ 0000_H$

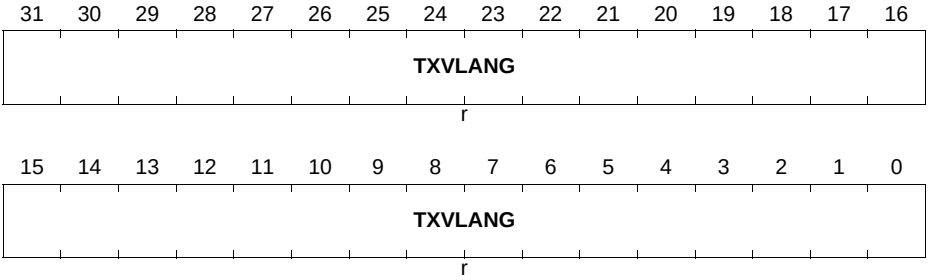


Field	Bits	Type	Description
TXPAUSE	[31:0]	r	**TXPAUSE** This field indicates the number of transmitted good PAUSE frames.

32-bit Register - Tx_VLAN_Frames_Good

This register maintains the number of transmitted good VLAN frames, exclusive of retried frames.

ETH_TX_VLAN_FRAMES_GOOD
Register 93 - Transmit Frame Count for Good VLAN Frames (1174_H) Reset Value: $0000\ 0000_H$

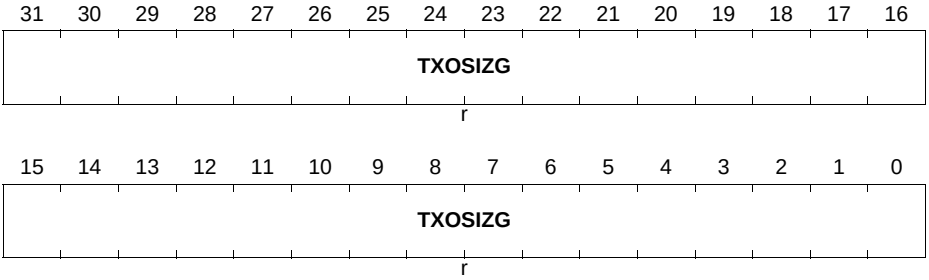


Field	Bits	Type	Description
TXVLANG	[31:0]	r	**TXVLANG** This register maintains the number of transmitted good VLAN frames, exclusive of retried frames.

32-bit Register - Tx_OSize_Frames_Good

This register maintains the number of transmitted good Oversize frames, exclusive of retried frames.

ETH_TX_OSIZE_FRAMES_GOOD
Register 94 - Transmit Frame Count for Good Oversize Frames (1178_H) Reset Value: $0000\ 0000_H$

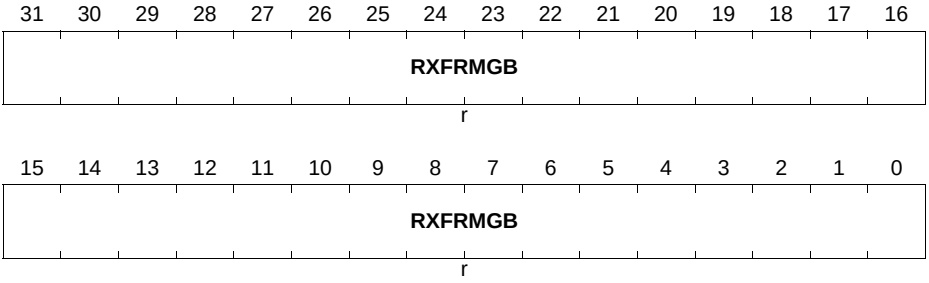


Field	Bits	Type	Description
TXOSIZG	[31:0]	r	**TXOSIZG** This field indicates the number of frames transmitted without errors and with length greater than the maxsize (1,518 or 1,522 bytes for VLAN tagged frames; 2000 bytes if enabled in bit 27 of Register 0 (MAC Configuration Register)).

32-bit Register - Rx_Frames_Count_Good_Bad

This register maintains the number of received good and bad frames.

ETH_RX_FRAMES_COUNT_GOOD_BAD
Register 96 - Receive Frame Count for Good and Bad Frames (1180_H) Reset Value: $0000\ 0000_H$

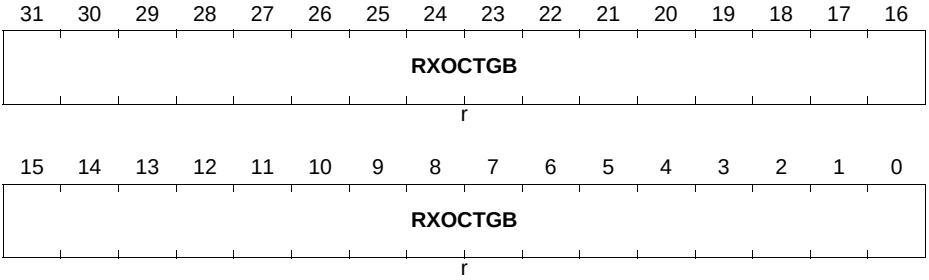


Field	Bits	Type	Description
RXFRMGB	[31:0]	r	**RXFRMGB** This field indicates the number of received good and bad frames.

32-bit Register - Rx_Octet_Count_Good_Bad

This register maintains the number of bytes received, exclusive of preamble, in good and bad frames.

ETH_RX_OCTET_COUNT_GOOD_BAD
Register 97 - Receive Octet Count for Good and Bad Frames (1184_H) Reset Value: $0000\ 0000_H$



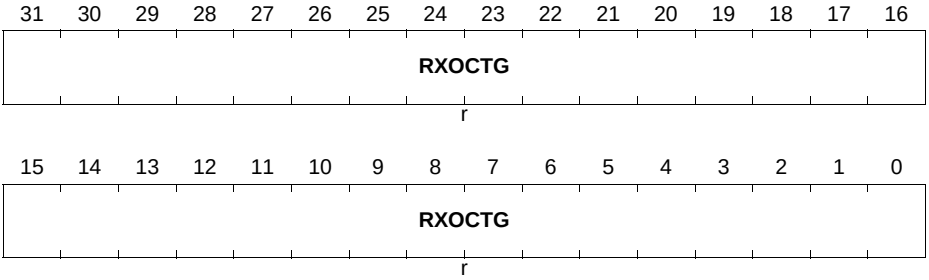
Field	Bits	Type	Description
RXOCTGB	[31:0]	r	**RXOCTGB** This field indicates the number of bytes received, exclusive of preamble, in good and bad frames.

32-bit Register - Rx_Octet_Count_Good

This register maintains the number of bytes received, exclusive of preamble, only in good frames.

ETH_RX_OCTET_COUNT_GOOD

Register 98 - Receive Octet Count for Good Frames (1188_H) Reset Value: $0000\ 0000_H$

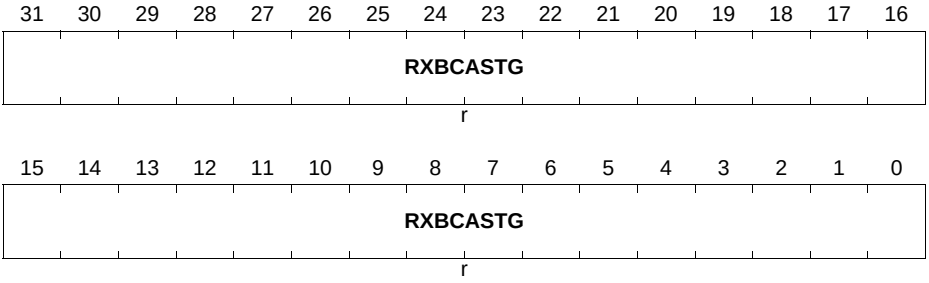


Field	Bits	Type	Description
RXOCTG	[31:0]	r	**RXOCTG** This field indicates the number of bytes received, exclusive of preamble, only in good frames.

32-bit Register - Rx_Broadcast_Frames_Good

This register maintains the number of received good broadcast frames.

ETH_RX_BROADCAST_FRAMES_GOOD
Register 99 - Receive Frame Count for Good Broadcast Frames ($118C_H$) Reset Value: $0000\ 0000_H$

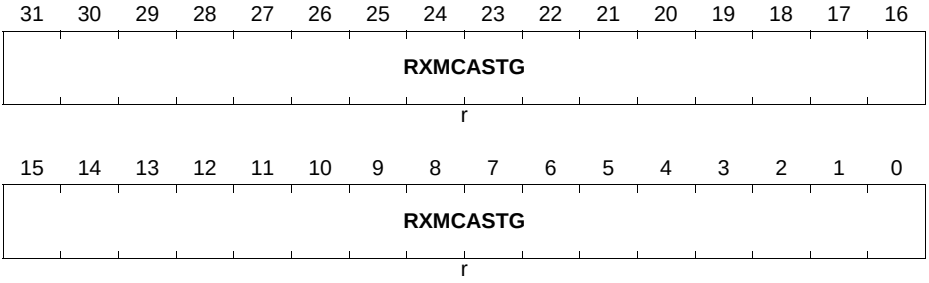


Field	Bits	Type	Description
RXBCASTG	[31:0]	r	**RXBCASTG** This field indicates the number of received good broadcast frames.

32-bit Register - Rx_Multicast_Frames_Good

This register maintains the number of received good multicast frames.

ETH_RX_MULTICAST_FRAMES_GOOD
Register 100 - Receive Frame Count for Good Multicast Frames (1190_H) Reset Value: $0000\ 0000_H$



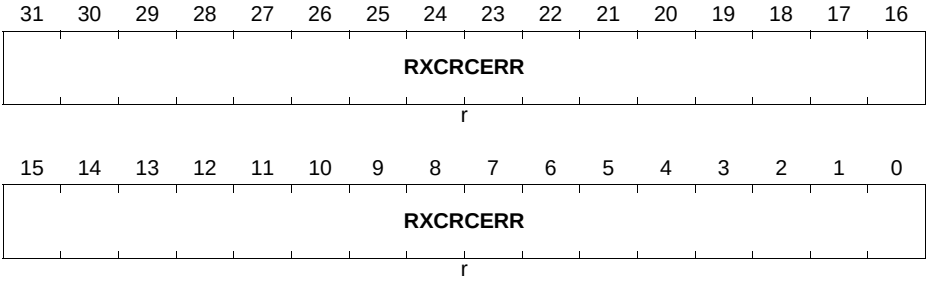
Field	Bits	Type	Description
RXMCASTG	[31:0]	r	**RXMCASTG** This field indicates the number of received good multicast frames.

32-bit Register - Rx_CRC_Error_Frames

This register maintains the number of frames received with CRC error.

ETH_RX_CRC_ERROR_FRAMES

Register 101 - Receive Frame Count for CRC Error Frames (1194_H) Reset Value: $0000\ 0000_H$

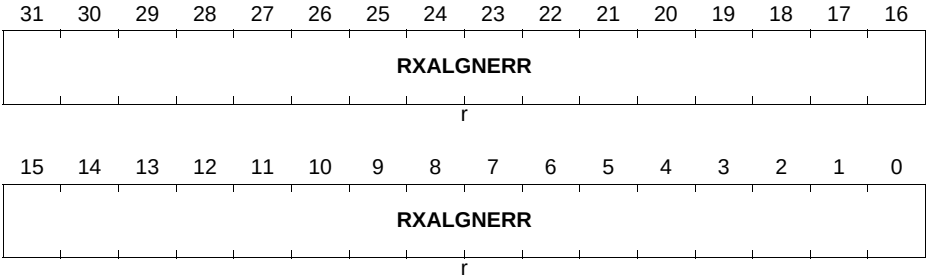


Field	Bits	Type	Description
RXCRCERR	[31:0]	r	**RXCRCERR** This field indicates the number of frames received with CRC error.

32-bit Register - Rx_Alignment_Error_Frames

This register maintains the number of frames received with alignment (dribble) error. This field is valid only in the 10 or 100 Mbps mode.

ETH_RX_ALIGNMENT_ERROR_FRAMES
Register 102 - Receive Frame Count for Alignment Error Frames (1198_H) Reset Value: $0000\ 0000_H$

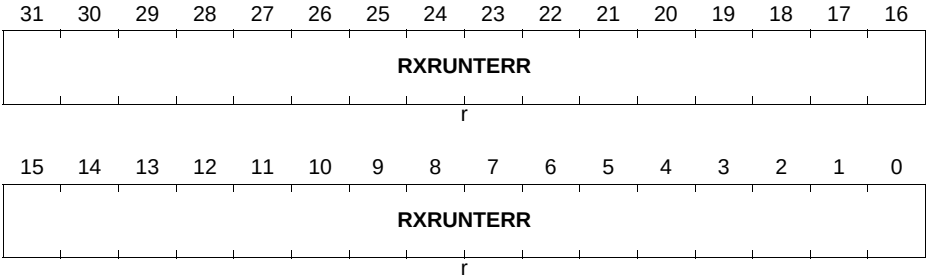


Field	Bits	Type	Description
RXALGNERR	[31:0]	r	**RXALGNERR** This field indicates the number of frames received with alignment (dribble) error. This field is valid only in the 10 or 100 Mbps mode.

32-bit Register - Rx_Runt_Error_Frames

This register maintains the number of frames received with runt error(<64 bytes and CRC error).

ETH_RX_RUNT_ERROR_FRAMES
Register 103 - Receive Frame Count for Runt Error Frames ($119C_H$) Reset Value: $0000\ 0000_H$



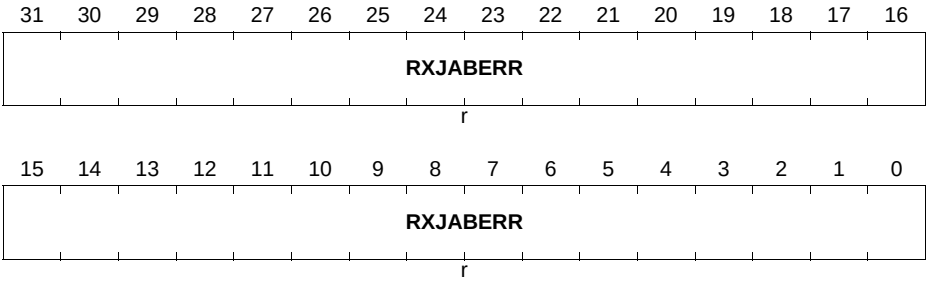
Field	Bits	Type	Description
RXRUNTERR	[31:0]	r	**RXRUNTERR** This field indicates the number of frames received with runt error(<64 bytes and CRC error).

32-bit Register - Rx_Jabber_Error_Frames

This register maintains the number of giant frames received with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If Jumbo Frame mode is enabled, then frames of length greater than 9,018 bytes (9,022 for VLAN tagged) are considered as giant frames.

ETH_RX_JABBER_ERROR_FRAMES

Register 104 - Receive Frame Count for Jabber Error Frames ($11A0_H$) Reset Value: $0000\ 0000_H$

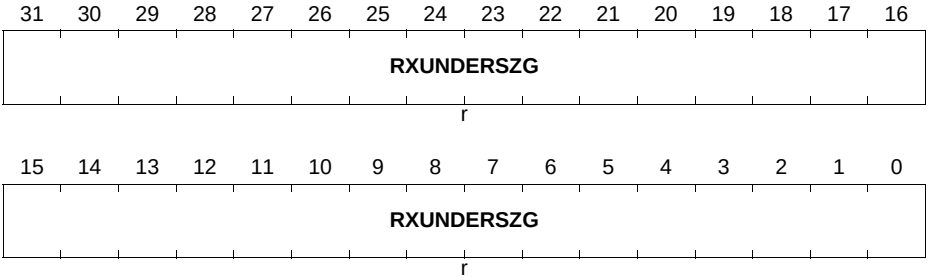


Field	Bits	Type	Description
RXJABERR	[31:0]	r	**RXJABERR** This field indicates the number of giant frames received with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If Jumbo Frame mode is enabled, then frames of length greater than 9,018 bytes (9,022 for VLAN tagged) are considered as giant frames.

32-bit Register - Rx_Undersize_Frames_Good

This register maintains the number of frames received with length less than 64 bytes and without errors.

ETH_RX_UNDERSIZE_FRAMES_GOOD
Register 105 - Receive Frame Count for Undersize Frames ($11A4_H$) Reset Value: $0000\ 0000_H$

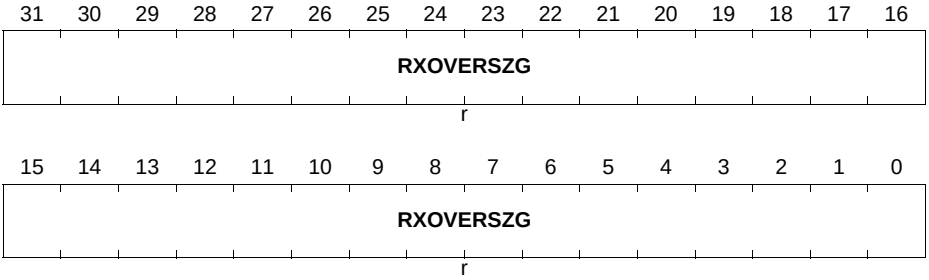


Field	Bits	Type	Description
RXUNDERSZG	[31:0]	r	**RXUNDERSZG** This field indicates the number of frames received with length less than 64 bytes and without errors.

32-bit Register - Rx_Oversize_Frames_Good

This register maintains the number of frames received with length greater than the maxsize (1,518 or 1,522 for VLAN tagged frames) and without errors.

ETH_RX_OVERSIZE_FRAMES_GOOD
Register 106 - Receive Frame Count for Oversize Frames ($11A8_H$) Reset Value: $0000\ 0000_H$

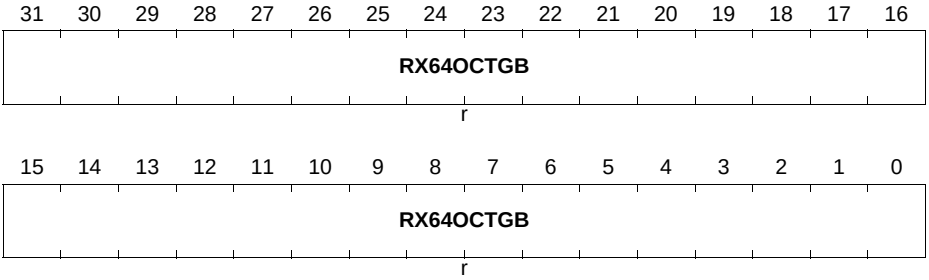


Field	Bits	Type	Description
RXOVERSZG	[31:0]	r	**RXOVERSZG** This field indicates the number of frames received without errors, with length greater than the maxsize (1,518 or 1,522 for VLAN tagged frames; 2,000 bytes if enabled in bit 27 of Register 0 (MAC Configuration Register)).

32-bit Register - Rx_64Octets_Frames_Good_Bad

This register maintains the number of received good and bad frames with length 64 bytes, exclusive of preamble.

ETH_RX_64OCTETS_FRAMES_GOOD_BAD
Register 107 - Receive Frame Count for Good and Bad 64 Byte Frames ($11AC_H$)
Reset Value: $0000\ 0000_H$

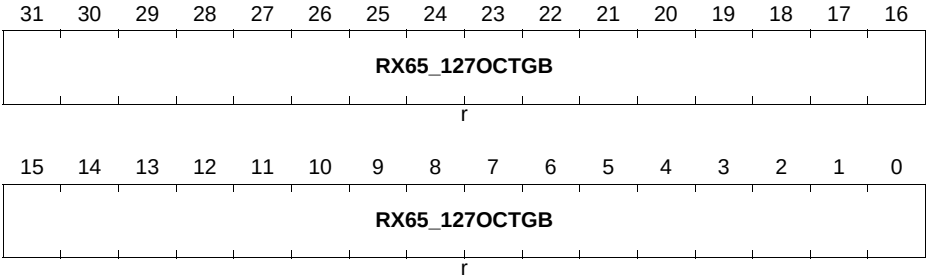


Field	Bits	Type	Description
RX64OCTGB	[31:0]	r	**RX64OCTGB** This field indicates the number of received good and bad frames with length 64 bytes, exclusive of preamble.

32-bit Register - Rx_65To127Octets_Frames_Good_Bad

This register maintains the number of received good and bad frames received with length between 65 and 127 (inclusive) bytes, exclusive of preamble.

ETH_RX_65TO127OCTETS_FRAMES_GOOD_BAD
Register 108 - Receive Frame Count for Good and Bad 65 to 127 Bytes Frames ($11B0_H$) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RX65_127 OCTGB	[31:0]	r	**RX65_127OCTGB** This field indicates the number of received good and bad frames received with length between 65 and 127 (inclusive) bytes, exclusive of preamble.

32-bit Register - Rx_128To255Octets_Frames_Good_Bad

This register maintains the number of received good and bad frames with length between 128 and 255 (inclusive) bytes, exclusive of preamble.

ETH_RX_128TO255OCTETS_FRAMES_GOOD_BAD
Register 109 - Receive Frame Count for Good and Bad 128 to 255 Bytes Frames ($11B4_H$) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RX128_255OCTGB	[31:0]	r	**RX128_255OCTGB** This field indicates the number of received good and bad frames with length between 128 and 255 (inclusive) bytes, exclusive of preamble.

32-bit Register - Rx_256To511Octets_Frames_Good_Bad

This register maintains the number of received good and bad frames with length between 256 and 511 (inclusive) bytes, exclusive of preamble.

ETH_RX_256TO511OCTETS_FRAMES_GOOD_BAD
Register 110 - Receive Frame Count for Good and Bad 256 to 511 Bytes Frames ($11B8_H$) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RX256_511OCTGB	[31:0]	r	**RX256_511OCTGB** This field indicates the number of received good and bad frames with length between 256 and 511 (inclusive) bytes, exclusive of preamble.

32-bit Register - Rx_512To1023Octets_Frames_Good_Bad

This register maintains the number of received good and bad frames with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble.

ETH_RX_512TO1023OCTETS_FRAMES_GOOD_BAD
Register 111 - Receive Frame Count for Good and Bad 512 to 1,023 Bytes Frames ($11BC_H$) **Reset Value: $0000\ 0000_H$**

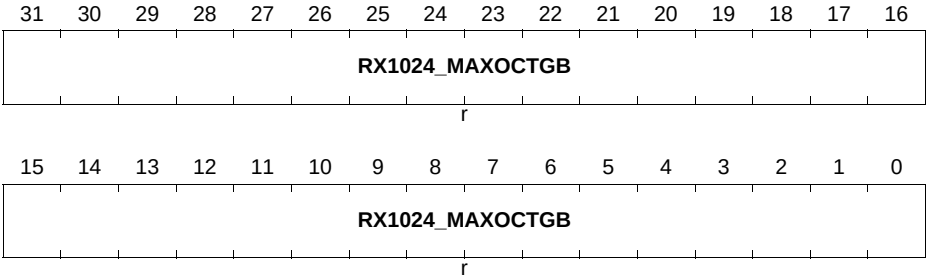


Field	Bits	Type	Description
RX512_1023OCTGB	[31:0]	r	**RX512_1023OCTGB** This field indicates the number of received good and bad frames with length between 512 and 1,023 (inclusive) bytes, exclusive of preamble.

32-bit Register - Rx_1024ToMaxOctets_Frames_Good_Bad

This register maintains the number of received good and bad frames with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble.

ETH_RX_1024TOMAXOCTETS_FRAMES_GOOD_BAD
Register 112 - Receive Frame Count for Good and Bad 1,024 to Maxsize Bytes Frames **($11C0_H$)** **Reset Value: $0000\ 0000_H$**

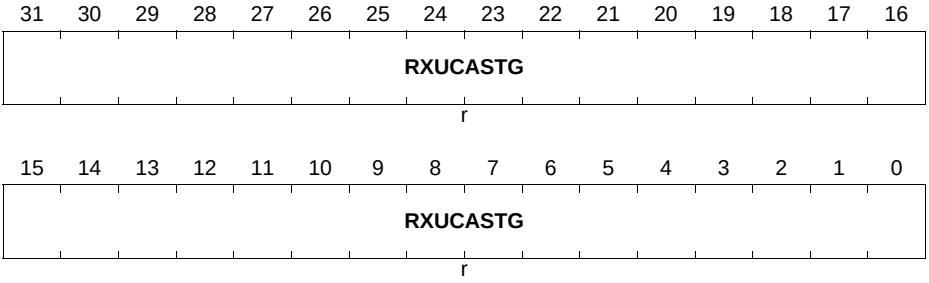


Field	Bits	Type	Description
RX1024_MAXOCTGB	[31:0]	r	**RX1024_MAXOCTGB** This field indicates the number of received good and bad frames with length between 1,024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames.

32-bit Register - Rx_Unicast_Frames_Good

This register maintains the number of received good unicast frames.

ETH_RX_UNICAST_FRAMES_GOOD
Register 113 - Receive Frame Count for Good Unicast Frames ($11C4_H$) Reset Value: $0000\ 0000_H$

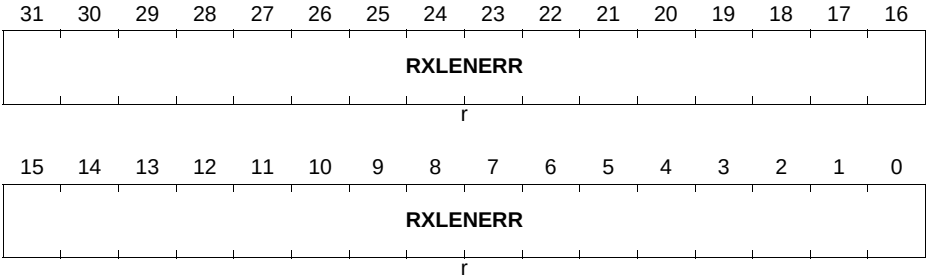


Field	Bits	Type	Description
RXUCASTG	[31:0]	r	**RXUCASTG** This field indicates the number of received good unicast frames.

32-bit Register - Rx_Length_Error_Frames

This register maintains the number of frames received with length error (Length type field not equal to frame size) for all frames with valid length field.

ETH_RX_LENGTH_ERROR_FRAMES
Register 114 - Receive Frame Count for Length Error Frames ($11C8_H$) Reset Value: $0000\ 0000_H$

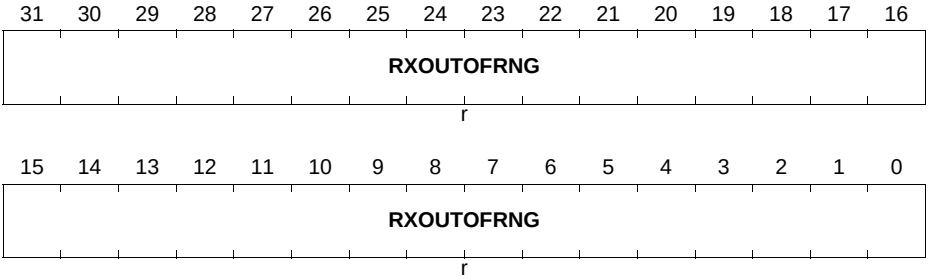


Field	Bits	Type	Description
RXLENERR	[31:0]	r	**RXLENERR** This field indicates the number of frames received with length error (Length type field not equal to frame size) for all frames with valid length field.

32-bit Register - Rx_Out_Of_Range_Type_Frames

This register maintains the number of received frames with length field not equal to the valid frame size (greater than 1,500 but less than 1,536).

ETH_RX_OUT_OF_RANGE_TYPE_FRAMES
Register 115 - Receive Frame Count for Out of Range Frames ($11CC_H$) Reset Value: $0000\ 0000_H$



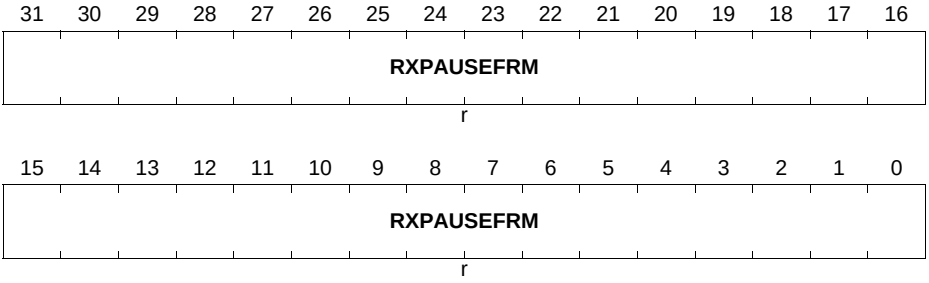
Field	Bits	Type	Description
RXOUTOFRNG	[31:0]	r	**RXOUTOFRNG** This field indicates the number of received frames with length field not equal to the valid frame size (greater than 1,500 but less than 1,536).

32-bit Register - Rx_Pause_Frames

This register maintains the number of received good and valid PAUSE frames.

ETH_RX_PAUSE_FRAMES

Register 116 - Receive Frame Count for PAUSE Frames ($11D0_H$)Reset Value: $0000\ 0000_H$

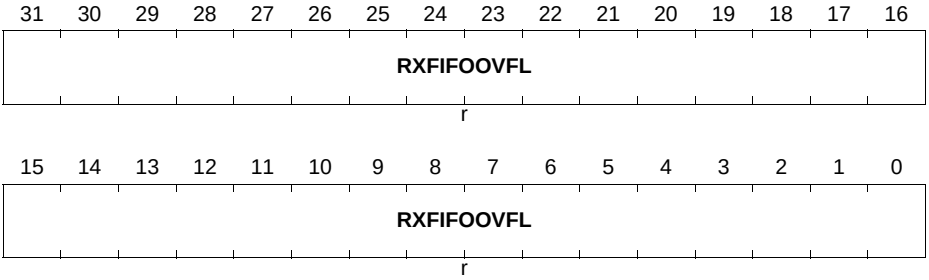


Field	Bits	Type	Description
RXPAUSEFRM	[31:0]	r	**RXPAUSEFRM** This field indicates the number of received good and valid PAUSE frames.

32-bit Register - Rx_FIFO_Overflow_Frames

This register maintains the number of received frames missed because of FIFO overflow.

ETH_RX_FIFO_OVERFLOW_FRAMES
Register 117 - Receive Frame Count for FIFO Overflow Frames ($11D4_H$) Reset Value: $0000\ 0000_H$

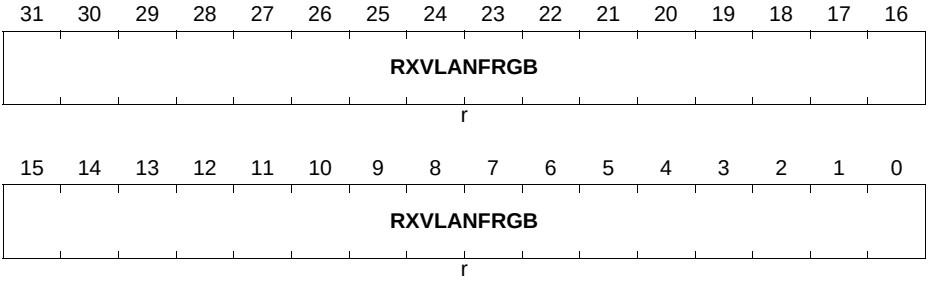


Field	Bits	Type	Description
RXFIFOOVFL	[31:0]	r	**RXFIFOOVFL** This field indicates the number of received frames missed because of FIFO overflow.

32-bit Register - Rx_VLAN_Frames_Good_Bad

This register maintains the number of received good and bad VLAN frames.

ETH_RX_VLAN_FRAMES_GOOD_BAD
Register 118 - Receive Frame Count for Good and Bad VLAN Frames ($11D8_H$)
Reset Value: $0000\ 0000_H$

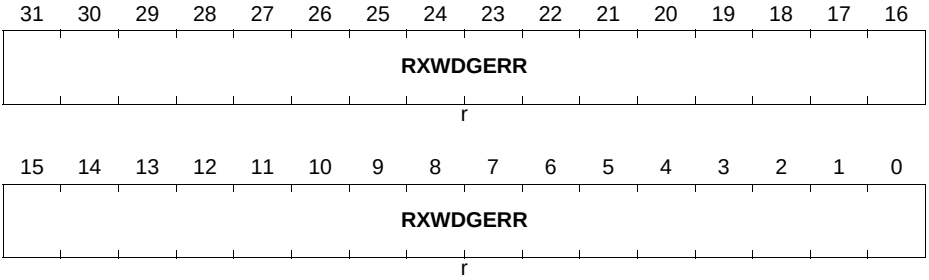


Field	Bits	Type	Description
RXVLANFRGB	[31:0]	r	**RXVLANFRGB** This field indicates the number of received good and bad VLAN frames.

32-bit Register - Rx_Watchdog_Error_Frames

This register maintains the number of frames received with error because of the watchdog timeout error (frames with more than 2,048 bytes data load).

ETH_RX_WATCHDOG_ERROR_FRAMES
Register 119 - Receive Frame Count for Watchdog Error Frames ($11DC_H$) Reset Value: $0000\ 0000_H$

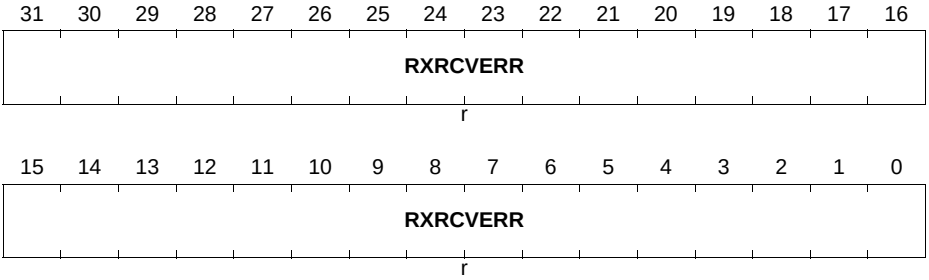


Field	Bits	Type	Description
RXWDGERR	[31:0]	r	**RXWDGERR** This field indicates the number of frames received with error because of the watchdog timeout error (frames with more than 2,048 bytes data load).

32-bit Register - Rx_Receive_Error_Frames

This register maintains the number of frames received with error because of the GMII/MII RXER error.

ETH_RX_RECEIVE_ERROR_FRAMES
Register 120 - Receive Frame Count for Receive Error Frames ($11E0_H$) Reset Value: $0000\ 0000_H$

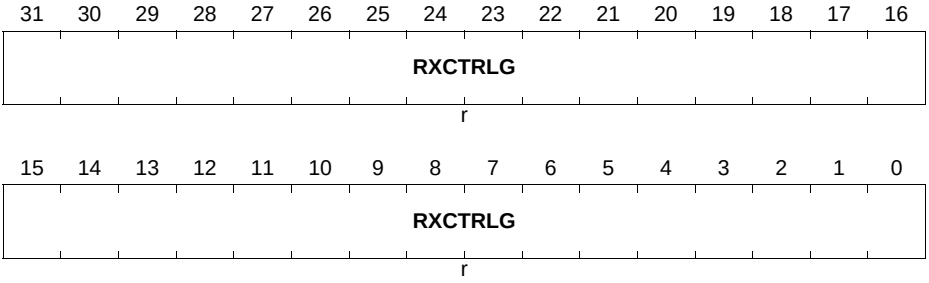


Field	Bits	Type	Description
RXRCVERR	[31:0]	r	**RXRCVERR** This field indicates the number of frames received with error because of the watchdog timeout error (frames with more than 2,048 bytes data load).

32-bit Register - Rx_Control_Frames_Good

This register maintains the number of godd control frames received.

ETH_RX_CONTROL_FRAMES_GOOD
Register 121 - Receive Frame Count for Good Control Frames Frames ($11E4_H$)
Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
RXCTRLG	[31:0]	r	**RXCTRLG** This field indicates the number of frames received with error because of the watchdog timeout error (frames with more than 2,048 bytes data load).

32-bit Register - MMC_IPC_Receive_Interrupt_Mask

This register maintains the mask for the interrupt generated from the receive IPC statistic counters. This register is 32-bits wide. This register is present only when any one of the MMC Receive IPC Counters is selected during core configuration.

ETH_MMC_IPC_RECEIVE_INTERRUPT_MASK
Register 128 - MMC Receive Checksum Offload Interrupt Mask Register (1200_H)
Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED_31_30		RXICMPEROIM	RXICMPGOIM	RXTCPEROIM	RXTCPGOIM	RXUDPEROIM	RXUDPGOIM	RXIPV6NOPAYOIM	RXIPV6HEROIM	RXIPV6GOIM	RXIPV4UDSBLOIM	RXIPV4FRAGOIM	RXIPV4NOPAYOIM	RXIPV4HEROIM	RXIPV4GOIM
r		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED_15_14		RXICMPERFIM	RXICMPGFIM	RXTCPERFIM	RXTCPGFIM	RXUDPERFIM	RXUDPGFIM	RXIPV6NOPAYFIM	RXIPV6HERFIM	RXIPV6GFIM	RXIPV4UDSBLFIM	RXIPV4FRAGFIM	RXIPV4NOPAYFIM	RXIPV4HERFIM	RXIPV4GFIM
r		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
RXIPV4GFIM	0	rw	**MMC Receive IPV4 Good Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv4_gd_frms counter reaches half of the maximum value or the maximum value.
RXIPV4HERFIM	1	rw	**MMC Receive IPV4 Header Error Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv4_hdrerr_frms counter reaches half of the maximum value or the maximum value.
RXIPV4NOPAYFIM	2	rw	**MMC Receive IPV4 No Payload Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv4_nopay_frms counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RXIPV4FRAGFIM	3	rw	**MMC Receive IPV4 Fragmented Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv4_frag_frms counter reaches half of the maximum value or the maximum value.
RXIPV4UDSBLFIM	4	rw	**MMC Receive IPV4 UDP Checksum Disabled Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv4_udsbl_frms counter reaches half of the maximum value or the maximum value.
RXIPV6GFIM	5	rw	**MMC Receive IPV6 Good Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv6_gd_frms counter reaches half of the maximum value or the maximum value.
RXIPV6HERFIM	6	rw	**MMC Receive IPV6 Header Error Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv6_hdrerr_frms counter reaches half of the maximum value or the maximum value.
RXIPV6NOPAYFIM	7	rw	**MMC Receive IPV6 No Payload Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv6_nopay_frms counter reaches half of the maximum value or the maximum value.
RXUDPGFIM	8	rw	**MMC Receive UDP Good Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxudp_gd_frms counter reaches half of the maximum value or the maximum value.
RXUDPERFIM	9	rw	**MMC Receive UDP Error Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxudp_err_frms counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RXTCPGFIM	10	rw	**MMC Receive TCP Good Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxtcp_gd_frms counter reaches half of the maximum value or the maximum value.
RXTCPERFIM	11	rw	**MMC Receive TCP Error Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxtcp_err_frms counter reaches half of the maximum value or the maximum value.
RXICMPGFIM	12	rw	**MMC Receive ICMP Good Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxicmp_gd_frms counter reaches half of the maximum value or the maximum value.
RXICMPERFIM	13	rw	**MMC Receive ICMP Error Frame Counter Interrupt Mask** Setting this bit masks the interrupt when the rxicmp_err_frms counter reaches half of the maximum value or the maximum value.
RESERVED_15_14	[15:14]	r	**RESERVED_15_14**
RXIPV4GOIM	16	rw	**MMC Receive IPV4 Good Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv4_gd_octets counter reaches half of the maximum value or the maximum value.
RXIPV4HEROIM	17	rw	**MMC Receive IPV4 Header Error Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv4_hdrerr_octets counter reaches half of the maximum value or the maximum value.
RXIPV4NOPAYOIM	18	rw	**MMC Receive IPV4 No Payload Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv4_nopay_octets counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RXIPV4FRAGOIM	19	rw	**MMC Receive IPV4 Fragmented Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv4_frag_octets counter reaches half of the maximum value or the maximum value.
RXIPV4UDSBLOIM	20	rw	**MMC Receive IPV4 UDP Checksum Disabled Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv4_udsbl_octets counter reaches half of the maximum value or the maximum value.
RXIPV6GOIM	21	rw	**MMC Receive IPV6 Good Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv6_gd_octets counter reaches half of the maximum value or the maximum value.
RXIPV6HEROIM	22	rw	**MMC Receive IPV6 Header Error Octet Counter Interrupt Mask** Setting this bit masks interrupt when the rxipv6_hdrerr_octets counter reaches half of the maximum value or the maximum value.
RXIPV6NOPAYOIM	23	rw	**MMC Receive IPV6 No Payload Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv6_nopay_octets counter reaches half of the maximum value or the maximum value.
RXUDPGOIM	24	rw	**MMC Receive UDP Good Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxudp_gd_octets counter reaches half of the maximum value or the maximum value.
RXUDPEROIM	25	rw	**MMC Receive UDP Error Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxudp_err_octets counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RXTCPGOIM	26	rw	**MMC Receive TCP Good Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxtcp_gd_octets counter reaches half of the maximum value or the maximum value.
RXTCPERROIM	27	rw	**MMC Receive TCP Error Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxtcp_err_octets counter reaches half of the maximum value or the maximum value.
RXICMPGOIM	28	rw	**MMC Receive ICMP Good Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxicmp_gd_octets counter reaches half of the maximum value or the maximum value.
RXICMPEROIM	29	rw	**MMC Receive ICMP Error Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxicmp_err_octets counter reaches half of the maximum value or the maximum value.
RESERVED_31_30	[31:30]	r	**RESERVED_31_30**

32-bit Register - MMC_IPC_Receive_Interrupt

This register maintains the interrupt that the receive IPC statistic counters generate.

ETH_MMC_IPC_RECEIVE_INTERRUPT
Register 130 - MMC Receive Checksum Offload Interrupt Register (1208_H) Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED_31_30		RXICMPEROIS	RXICMPGOIS	RXTCPEROIS	RXTCPGOIS	RXUDPEROIS	RXUDPGOIS	RXIPV6NOPAYOIS	RXIPV6HEROIS	RXIPV6GOIS	RXIPV4UDSBLOIS	RXIPV4FRAGOIS	RXIPV4NOPAYOIS	RXIPV4HEROIS	RXIPV4GOIS
r		r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED_15_14		RXICMPERFIS	RXICMPGFIS	RXTCPERFIS	RXTCPGFIS	RXUDPERFIS	RXUDPGFIS	RXIPV6NOPAYFIS	RXIPV6HERFIS	RXIPV6GFIS	RXIPV4UDSBLFIS	RXIPV4FRAGFIS	RXIPV4NOPAYFIS	RXIPV4HERFIS	RXIPV4GFIS
r		r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
RXIPV4GFIS	0	r	**MMC Receive IPV4 Good Frame Counter Interrupt Status** This bit is set when the rxipv4_gd_frms counter reaches half of the maximum value or the maximum value.
RXIPV4HERFIS	1	r	**MMC Receive IPV4 Header Error Frame Counter Interrupt Status** This bit is set when the rxipv4_hdrerr_frms counter reaches half of the maximum value or the maximum value.
RXIPV4NOPAYFIS	2	r	**MMC Receive IPV4 No Payload Frame Counter Interrupt Status** This bit is set when the rxipv4_nopay_frms counter reaches half of the maximum value or the maximum value.
RXIPV4FRAGFIS	3	r	**MMC Receive IPV4 Fragmented Frame Counter Interrupt Status** This bit is set when the rxipv4_frag_frms counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RXIPV4UDSBLFIS	4	r	**MMC Receive IPV4 UDP Checksum Disabled Frame Counter Interrupt Status** This bit is set when the rxipv4_udsbl_frms counter reaches half of the maximum value or the maximum value.
RXIPV6GFIS	5	r	**MMC Receive IPV6 Good Frame Counter Interrupt Status** This bit is set when the rxipv6_gd_frms counter reaches half of the maximum value or the maximum value.
RXIPV6HERFIS	6	r	**MMC Receive IPV6 Header Error Frame Counter Interrupt Status** This bit is set when the rxipv6_hdrerr_frms counter reaches half of the maximum value or the maximum value.
RXIPV6NOPAYFIS	7	r	**MMC Receive IPV6 No Payload Frame Counter Interrupt Status** This bit is set when the rxipv6_nopay_frms counter reaches half of the maximum value or the maximum value.
RXUDPGFIS	8	r	**MMC Receive UDP Good Frame Counter Interrupt Status** This bit is set when the rxudp_gd_frms counter reaches half of the maximum value or the maximum value.
RXUDPERFIS	9	r	**MMC Receive UDP Error Frame Counter Interrupt Status** This bit is set when the rxudp_err_frms counter reaches half of the maximum value or the maximum value.
RXTCPGFIS	10	r	**MMC Receive TCP Good Frame Counter Interrupt Status** This bit is set when the rxtcp_gd_frms counter reaches half of the maximum value or the maximum value.
RXTCPERFIS	11	r	**MMC Receive TCP Error Frame Counter Interrupt Status** This bit is set when the rxtcp_err_frms counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RXICMPGFIS	12	r	**MMC Receive ICMP Good Frame Counter Interrupt Status** This bit is set when the rxicmp_gd_frms counter reaches half of the maximum value or the maximum value.
RXICMPERFIS	13	r	**MMC Receive ICMP Error Frame Counter Interrupt Status** This bit is set when the rxicmp_err_frms counter reaches half of the maximum value or the maximum value.
RESERVED_15_14	[15:14]	r	**RESERVED_15_14**
RXIPV4GOIS	16	r	**MMC Receive IPV4 Good Octet Counter Interrupt Status** This bit is set when the rxipv4_gd_octets counter reaches half of the maximum value or the maximum value.
RXIPV4HEROIS	17	r	**MMC Receive IPV4 Header Error Octet Counter Interrupt Status** This bit is set when the rxipv4_hdrerr_octets counter reaches half of the maximum value or the maximum value.
RXIPV4NOPAYOIS	18	r	**MMC Receive IPV4 No Payload Octet Counter Interrupt Status** This bit is set when the rxipv4_nopay_octets counter reaches half of the maximum value or the maximum value.
RXIPV4FRAGOIS	19	r	**MMC Receive IPV4 Fragmented Octet Counter Interrupt Status** This bit is set when the rxipv4_frag_octets counter reaches half of the maximum value or the maximum value.
RXIPV4UDSBLOIS	20	r	**MMC Receive IPV4 UDP Checksum Disabled Octet Counter Interrupt Status** This bit is set when the rxipv4_udsbl_octets counter reaches half of the maximum value or the maximum value.

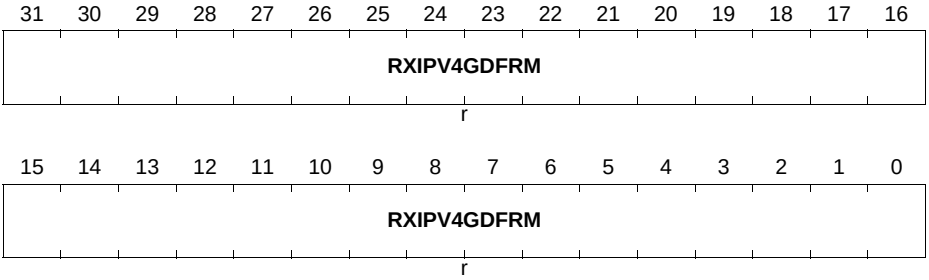
Field	Bits	Type	Description
RXIPV6GOIS	21	r	**MMC Receive IPV6 Good Octet Counter Interrupt Status** This bit is set when the rxipv6_gd_octets counter reaches half of the maximum value or the maximum value.
RXIPV6HEROIS	22	r	**MMC Receive IPV6 Header Error Octet Counter Interrupt Status** This bit is set when the rxipv6_hdrerr_octets counter reaches half of the maximum value or the maximum value.
RXIPV6NOPAYOIS	23	r	**MMC Receive IPV6 No Payload Octet Counter Interrupt Status** This bit is set when the rxipv6_nopay_octets counter reaches half of the maximum value or the maximum value.
RXUDPGOIS	24	r	**MMC Receive UDP Good Octet Counter Interrupt Status** This bit is set when the rxudp_gd_octets counter reaches half of the maximum value or the maximum value.
RXUDPEROIS	25	r	**MMC Receive UDP Error Octet Counter Interrupt Status** This bit is set when the rxudp_err_octets counter reaches half the maximum value or the maximum value.
RXTCPGOIS	26	r	**MMC Receive TCP Good Octet Counter Interrupt Status** This bit is set when the rxtcp_gd_octets counter reaches half the maximum value or the maximum value.
RXTCPEROIS	27	r	**MMC Receive TCP Error Octet Counter Interrupt Status** This bit is set when the rxtcp_err_octets counter reaches half of the maximum value or the maximum value.
RXICMPGOIS	28	r	**MMC Receive ICMP Good Octet Counter Interrupt Status** This bit is set when the rxicmp_gd_octets counter reaches half of the maximum value or the maximum value.

Field	Bits	Type	Description
RXICMPEROIS	29	r	**MMC Receive ICMP Error Octet Counter Interrupt Status** This bit is set when the rxicmp_err_octets counter reaches half of the maximum value or the maximum value.
RESERVED_31_30	[31:30]	r	**RESERVED_31_30**

32-bit Register - RxIPv4_Good_Frames

This register maintains the number of good IPv4 datagrams received with the TCP, UDP, or ICMP payload.

ETH_RXIPV4_GOOD_FRAMES
Register 132 - Receive IPV4 Good Frame Counter Register (1210_H) Reset Value: $0000\ 0000_H$

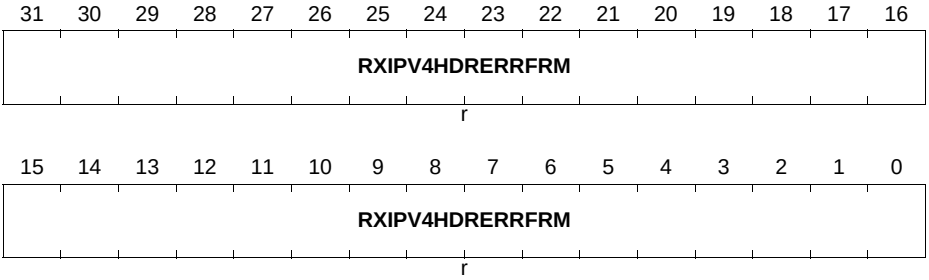


Field	Bits	Type	Description
RXIPV4GDFRM	[31:0]	r	**RXIPV4GDFRM** This field indicates the number of good IPv4 datagrams received with the TCP, UDP, or ICMP payload.

32-bit Register - RxIPv4_Header_Error_Frames

This register maintains the number of IPv4 datagrams received with header errors (checksum, length, or version mismatch).

ETH_RXIPV4_HEADER_ERROR_FRAMES
Register 133 - Receive IPV4 Header Error Frame Counter Register (1214_H) Reset Value: $0000\ 0000_H$

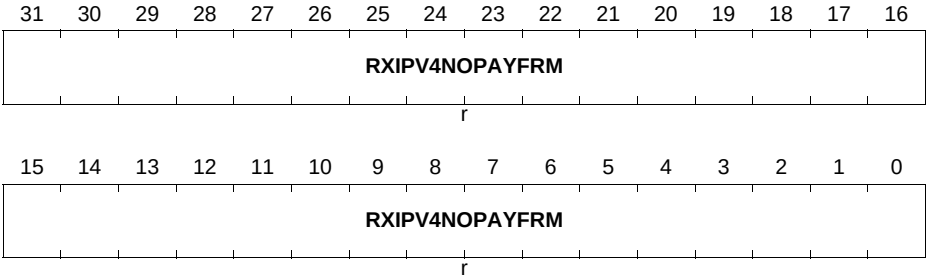


Field	Bits	Type	Description
RXIPV4HDRERRFRM	[31:0]	r	**RXIPV4HDRERRFRM** This field indicates the number of IPv4 datagrams received with header errors (checksum, length, or version mismatch).

32-bit Register - RxIPv4_No_Payload_Frames

This register maintains the number of received IPv4 datagram frames without a TCP, UDP, or ICMP payload processed by the Checksum engine.

ETH_RXIPV4_NO_PAYLOAD_FRAMES
Register 134 - Receive IPV4 No Payload Frame Counter Register (1218_H) Reset Value: $0000\ 0000_H$

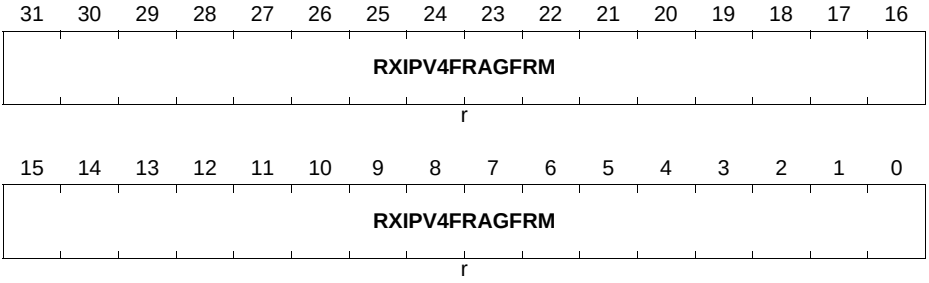


Field	Bits	Type	Description
RXIPV4NOPAYFRM	[31:0]	r	**RXIPV4NOPAYFRM** This field indicates the number of IPv4 datagram frames received that did not have a TCP, UDP, or ICMP payload processed by the Checksum engine.

32-bit Register - RxIPv4_Fragmented_Frames

This register maintains the number of good IPv4 datagrams received with fragmentation.

ETH_RXIPV4_FRAGMENTED_FRAMES
Register 135 - Receive IPV4 Fragmented Frame Counter Register ($121C_H$) Reset Value: $0000\ 0000_H$

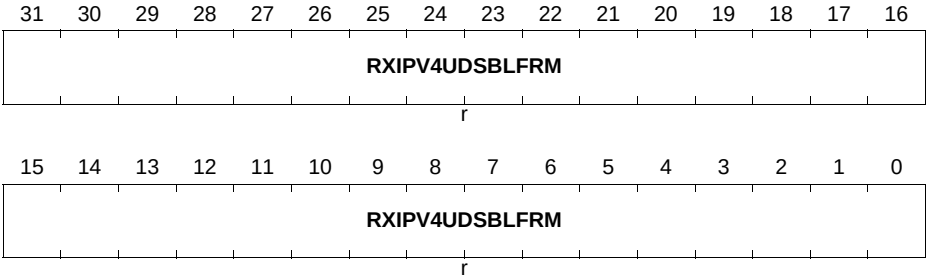


Field	Bits	Type	Description
RXIPV4FRAGFRM	[31:0]	r	**RXIPV4FRAGFRM** This field indicates the number of good IPv4 datagrams received with fragmentation.

32-bit Register - RxIPv4_UDP_Checksum_Disabled_Frames

This register maintains the number of received good IPv4 datagrams which have the UDP payload with checksum disabled.

ETH_RXIPV4_UDP_CHECKSUM_DISABLED_FRAMES
Register 136 - Receive IPV4 UDP Checksum Disabled Frame Counter Register (1220_H) **Reset Value: $0000\ 0000_H$**

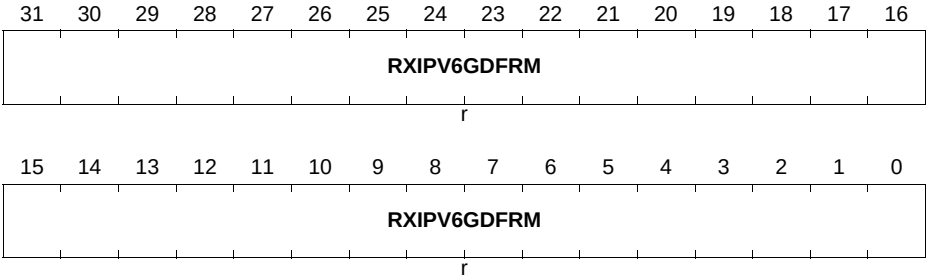


Field	Bits	Type	Description
RXIPV4UDSBLFRM	[31:0]	r	**RXIPV4UDSBLFRM** This field indicates the number of received good IPv4 datagrams which have the UDP payload with checksum disabled.

32-bit Register - RxIPv6_Good_Frames

This register maintains the number of good IPv6 datagrams received with TCP, UDP, or ICMP payloads.

ETH_RXIPV6_GOOD_FRAMES
Register 137 - Receive IPV6 Good Frame Counter Register (1224_H) Reset Value: $0000\ 0000_H$

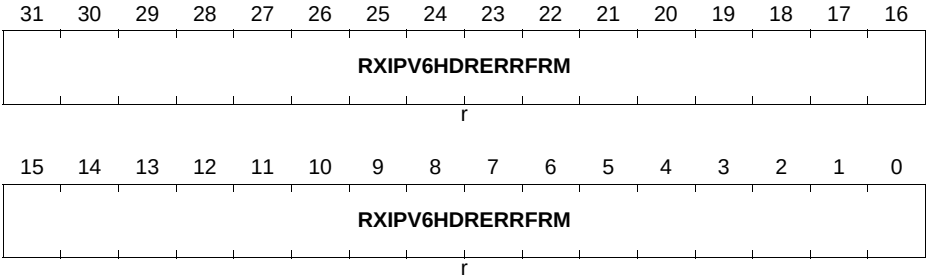


Field	Bits	Type	Description
RXIPV6GDFRM	[31:0]	r	**RXIPV6GDFRM** This field indicates the number of good IPv6 datagrams received with TCP, UDP, or ICMP payloads.

32-bit Register - RxIPv6_Header_Error_Frames

This register maintains the number of IPv6 datagrams received with header errors (length or version mismatch).

ETH_RXIPV6_HEADER_ERROR_FRAMES
Register 138 - Receive IPV6 Header Error Frame Counter Register (1228_H) Reset Value: $0000\ 0000_H$

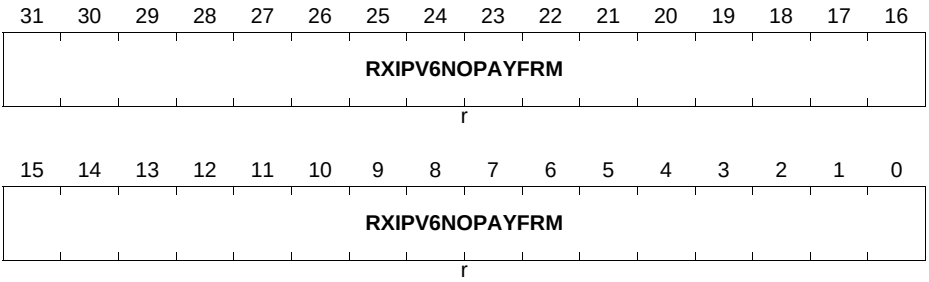


Field	Bits	Type	Description
RXIPV6HDRERRFRM	[31:0]	r	**RXIPV6HDRERRFRM** This field indicates the number of IPv6 datagrams received with header errors (length or version mismatch).

32-bit Register - RxIPv6_No_Payload_Frames

This register maintains the number of received IPv6 datagram frames without a TCP, UDP, or ICMP payload. This includes all IPv6 datagrams with fragmentation or security extension headers.

ETH_RXIPV6_NO_PAYLOAD_FRAMES
Register 139 - Receive IPV6 No Payload Frame Counter Register ($122C_H$) Reset Value: $0000\ 0000_H$

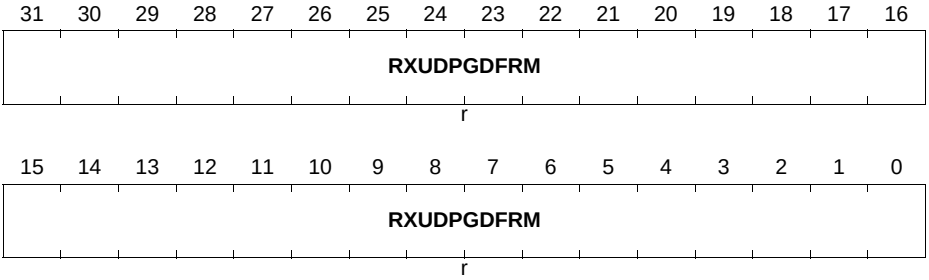


Field	Bits	Type	Description
RXIPV6NOPAYFRM	[31:0]	r	**RXIPV6NOPAYFRM** This field indicates the number of IPv6 datagram frames received that did not have a TCP, UDP, or ICMP payload. This includes all IPv6 datagrams with fragmentation or security extension headers.

32-bit Register - RxUDP_Good_Frames

This register maintains the number of good IP datagrams with a good UDP payload. This counter is not updated when the counter is incremented.

ETH_RXUDP_GOOD_FRAMES
Register 140 - Receive UDP Good Frame Counter Register (1230_H) Reset Value: $0000\ 0000_H$

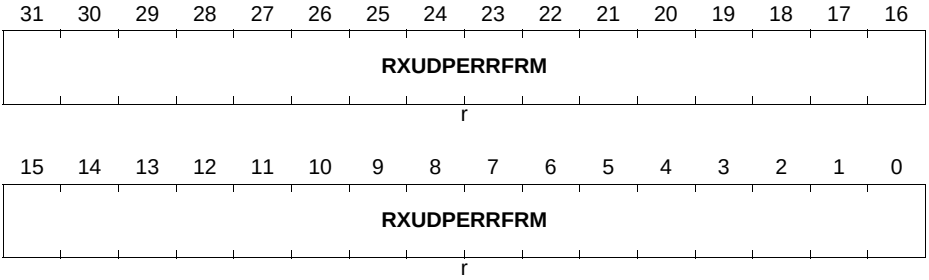


Field	Bits	Type	Description
RXUDPGDFRM	[31:0]	r	**RXUDPGDFRM** This field indicates the number of good IP datagrams with a good UDP payload. This counter is not updated when the counter is incremented.

32-bit Register - RxUDP_Error_Frames

This register maintains the number of good IP datagrams whose UDP payload has a checksum error.

ETH_RXUDP_ERROR_FRAMES
Register 141 - Receive UDP Error Frame Counter Register (1234_H) Reset Value: $0000\ 0000_H$

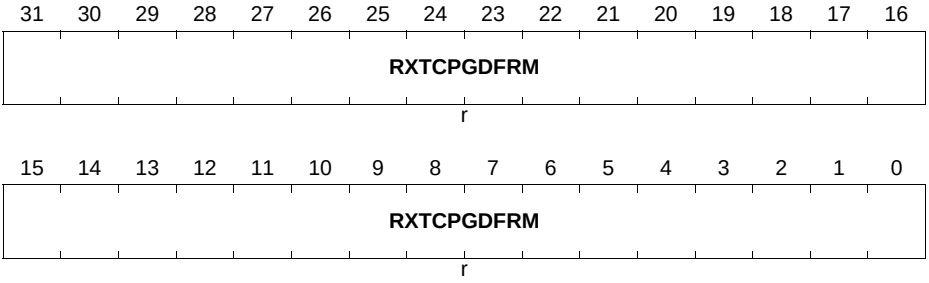


Field	Bits	Type	Description
RXUDPERRFRM	[31:0]	r	**RXUDPERRFRM** This field indicates the number of good IP datagrams whose UDP payload has a checksum error.

32-bit Register - RxTCP_Good_Frames

This register maintains the number of good IP datagrams with a good TCP payload.

ETH_RXTCP_GOOD_FRAMES
Register 142 - Receive TCP Good Frame Counter Register (1238_H) Reset Value: $0000\ 0000_H$

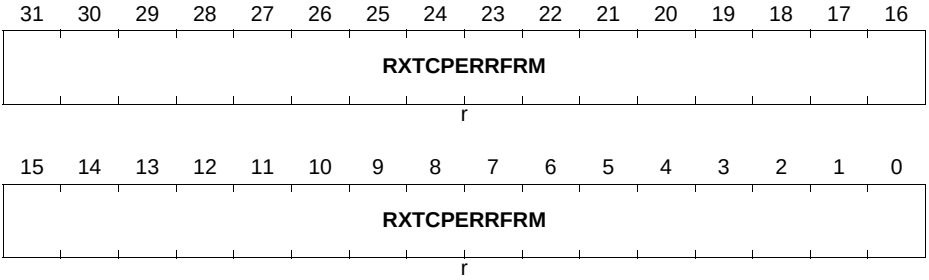


Field	Bits	Type	Description
RXTCPGDFRM	[31:0]	r	**RXTCPGDFRM** This field indicates the number of good IP datagrams with a good TCP payload.

32-bit Register - RxTCP_Error_Frames

This register maintains the number of good IP datagrams whose TCP payload has a checksum error.

ETH_RXTCP_ERROR_FRAMES
Register 143 - Receive TCP Error Frame Counter Register ($123C_H$) Reset Value: $0000\ 0000_H$

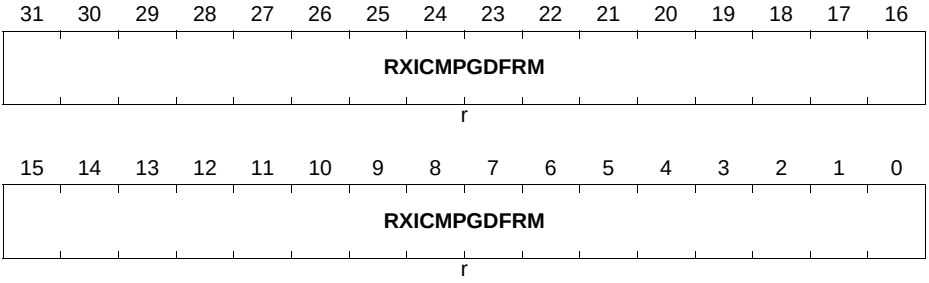


Field	Bits	Type	Description
RXTCPERRFRM	[31:0]	r	**RXTCPERRFRM** This field indicates the number of good IP datagrams whose TCP payload has a checksum error.

32-bit Register - RxICMP_Good_Frames

This register maintains the number of good IP datagrams with a good ICMP payload.

ETH_RXICMP_GOOD_FRAMES
Register 144 - Receive ICMP Good Frame Counter Register (1240_H) Reset Value: $0000\ 0000_H$

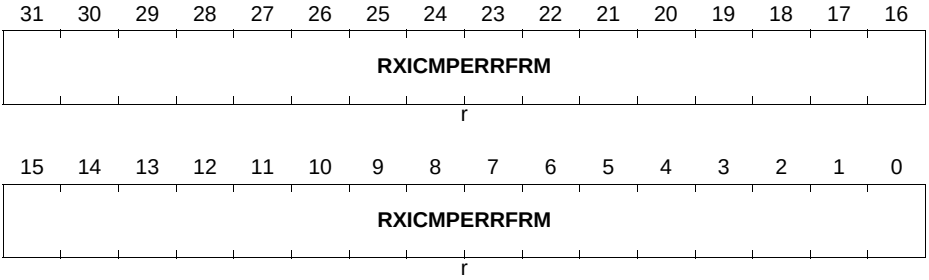


Field	Bits	Type	Description
RXICMPGDFRM	[31:0]	r	**RXICMPGDFRM** This field indicates the number of good IP datagrams with a good ICMP payload.

32-bit Register - RxICMP_Error_Frames

This register maintains the number of good IP datagrams whose ICMP payload has a checksum error.

ETH_RXICMP_ERROR_FRAMES
Register 145 - Receive ICMP Error Frame Counter Register (1244_H) Reset Value: $0000\ 0000_H$

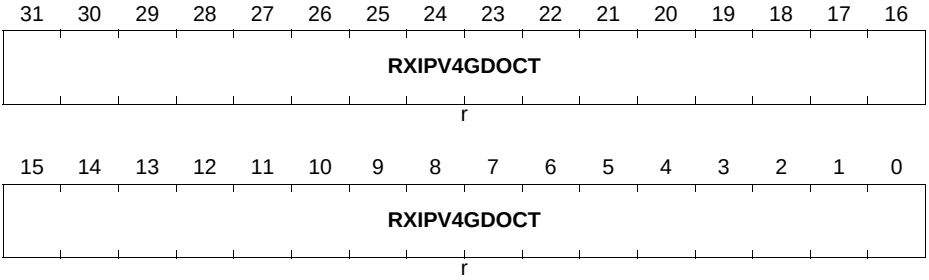


Field	Bits	Type	Description
RXICMPERRFRM	[31:0]	r	**RXICMPERRFRM** This field indicates the number of good IP datagrams whose ICMP payload has a checksum error.

32-bit Register - RxIPv4_Good_Octets

This register maintains the number of bytes received in good IPv4 datagrams encapsulating TCP, UDP, or ICMP data.

ETH_RXIPV4_GOOD_OCTETS
Register 148 - Receive IPV4 Good Octet Counter Register (1250_H) Reset Value: $0000\ 0000_H$

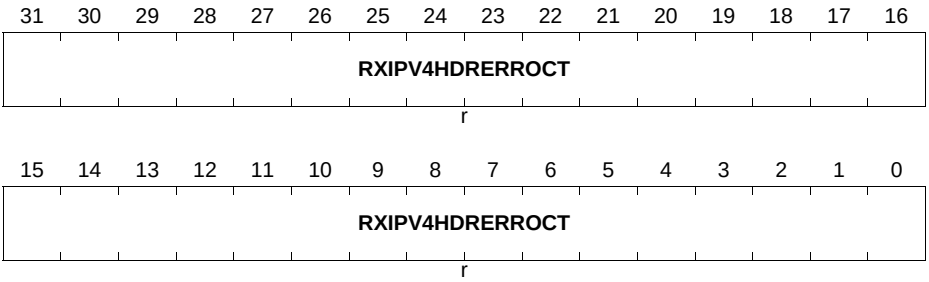


Field	Bits	Type	Description
RXIPV4GDOCT	[31:0]	r	**This field indicates the number of bytes received in good IPv4 datagrams encapsulating TCP, UDP, or ICMP data. The Ethernet header, FCS, pad, or IP pad** bytes are not included in this counter.

32-bit Register - RxIPv4_Header_Error_Octets

This register maintains the number of bytes received in IPv4 datagrams with header errors (checksum, length, or version mismatch). The value in the Length field of the IPv4 header is used to update this counter.

ETH_RXIPV4_HEADER_ERROR_OCTETS
Register 149 - Receive IPV4 Header Error Octet Counter Register (1254_H) Reset Value: $0000\ 0000_H$

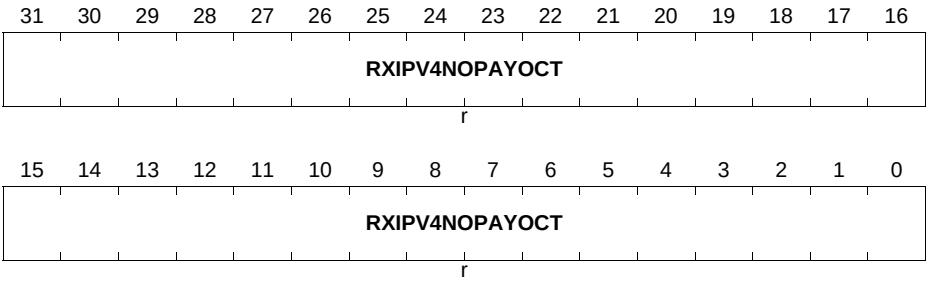


Field	Bits	Type	Description
RXIPV4HDRERROCT	[31:0]	r	**This field indicates the number of bytes received in the IPv4 datagrams with header errors (checksum, length, or version mismatch). The value in the Length field of IPv4 header is used to update this counter.** The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

32-bit Register - RxIPv4_No_Payload_Octets

This register maintains the number of bytes received in IPv4 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the IPv4 headers Length field is used to update this counter.

ETH_RXIPV4_NO_PAYLOAD_OCTETS
Register 150 - Receive IPV4 No Payload Octet Counter Register (1258_H) Reset Value: $0000\ 0000_H$

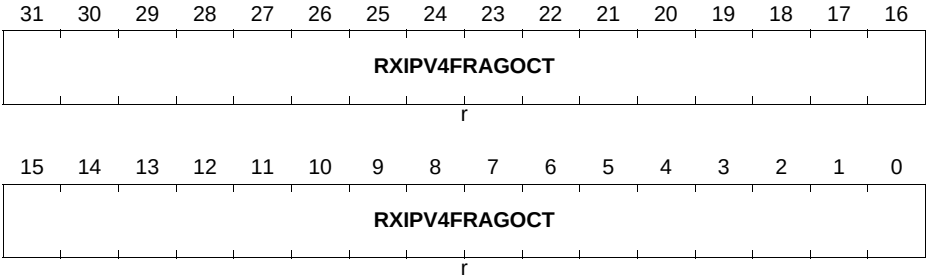


Field	Bits	Type	Description
RXIPV4NOPAYOCT	[31:0]	r	**This field indicates the number of bytes received in IPv4 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the IPv4 headers Length field is used to update this counter.** The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

32-bit Register - RxIPv4_Fragmented_Octets

This register maintains the number of bytes received in fragmented IPv4 datagrams. The value in the IPv4 headers Length field is used to update this counter.

ETH_RXIPV4_FRAGMENTED_OCTETS
Register 151 - Receive IPV4 Fragmented Octet Counter Register ($125C_H$) Reset Value: $0000\ 0000_H$

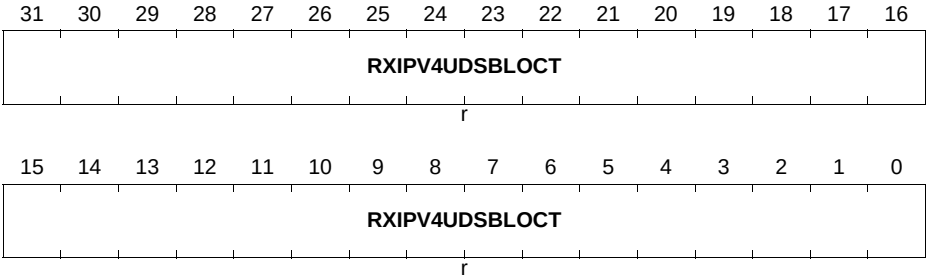


Field	Bits	Type	Description
RXIPV4FRAGOCT	[31:0]	r	**This field indicates the number of bytes received in fragmented IPv4 datagrams. The value in the IPv4 headers Length field is used to update this counter.** The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

32-bit Register - RxIPv4_UDP_Checksum_Disable_Octets

This register maintains the number of bytes received in a UDP segment that had the UDP checksum disabled.

ETH_RXIPV4_UDP_CHECKSUM_DISABLE_OCTETS
Register 152 - Receive IPV4 Fragmented Octet Counter Register (1260_H) Reset Value: $0000\ 0000_H$

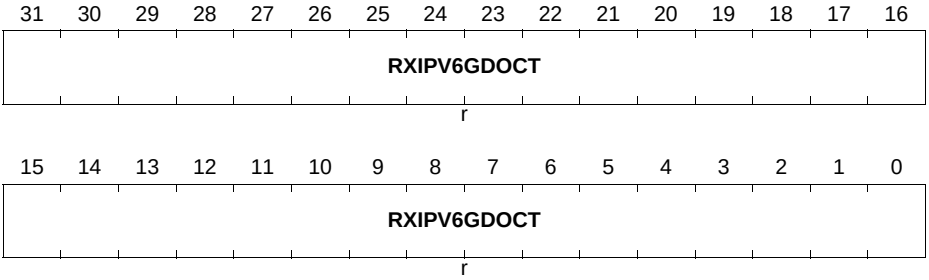


Field	Bits	Type	Description
RXIPV4UDSBLOCT	[31:0]	r	**RXIPV4UDSBLOCT** This field indicates the number of bytes received in a UDP segment that had the UDP checksum disabled. This counter does not count the IP Header bytes. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

32-bit Register - RxIPv6_Good_Octets

This register maintains the number of bytes received in good IPv6 datagrams encapsulating TCP, UDP or ICMPv6 data.

ETH_RXIPV6_GOOD_OCTETS
Register 153 - Receive IPV6 Good Octet Counter Register (1264_H) Reset Value: $0000\ 0000_H$

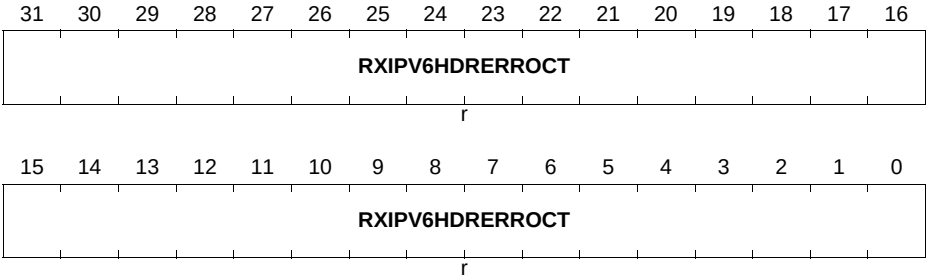


Field	Bits	Type	Description
RXIPV6GDOCT	[31:0]	r	**Thsi field indicates the number of bytes received in good IPv6 datagrams encapsulating TCP, UDP or ICMPv6 data.** This counter does not count the IP Header bytes. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

32-bit Register - RxIPv6_Header_Error_Octets

This register maintains the number of bytes received in IPv6 datagrams with header errors (length or version mismatch).

ETH_RXIPV6_HEADER_ERROR_OCTETS
Register 154 - Receive IPV6 Header Error Octet Counter Register (1268_H) Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
RXIPV6HDRERROCT	[31:0]	r	**This field indicates the number of bytes received in IPv6 datagrams with header errors (length or version mismatch). The value in the IPv6 headers Length field is used to update this counter.** This counter does not count the IP Header bytes. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

32-bit Register - RxIPv6_No_Payload_Octets

This register maintains the number of bytes received in IPv6 datagrams that did not have a TCP, UDP, or ICMP payload.

ETH_RXIPV6_NO_PAYLOAD_OCTETS
Register 155 - Receive IPV6 No Payload Octet Counter Register ($126C_H$) Reset Value: $0000\ 0000_H$

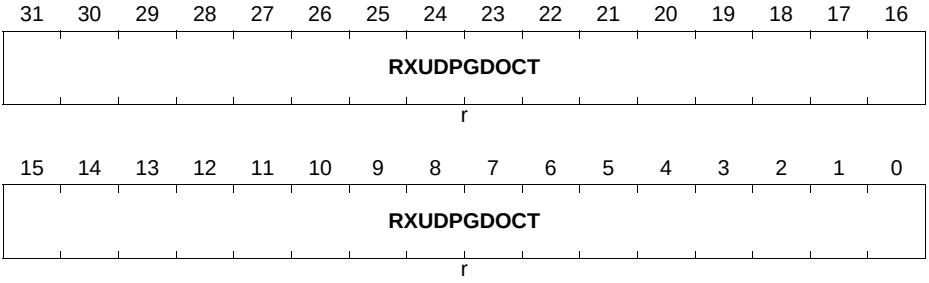


Field	Bits	Type	Description
RXIPV6NOPAYOCT	[31:0]	r	**This field indicates the number of bytes received in IPv6 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the IPv6 headers Length field is used to update this counter.** This counter does not count the IP Header bytes. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

32-bit Register - RxUDP_Good_Octets

This register maintains the number of bytes received in a good UDP segment.

ETH_RXUDP_GOOD_OCTETS
Register 156 - Receive UDP Good Octet Counter Register (1270_H) Reset Value: $0000\ 0000_H$

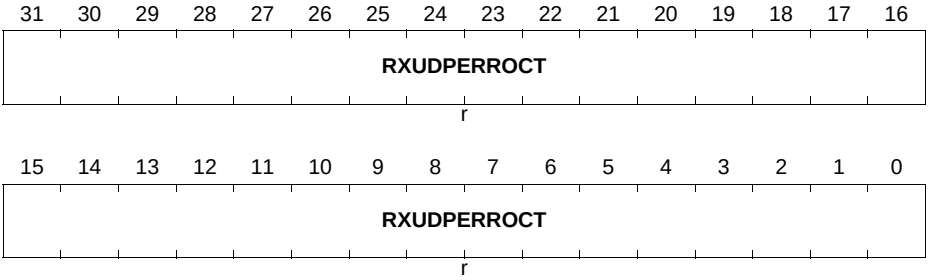


Field	Bits	Type	Description
RXUDPGDOCT	[31:0]	r	**RXUDPGDOCT** This field indicates the number of bytes received in a good UDP segment. This counter does not count IP header bytes. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

32-bit Register - RxUDP_Error_Octets

This register maintains the number of bytes received in a UDP segment with checksum errors.

ETH_RXUDP_ERROR_OCTETS
Register 157 - Receive UDP Error Octet Counter Register (1274_H) Reset Value: $0000\ 0000_H$

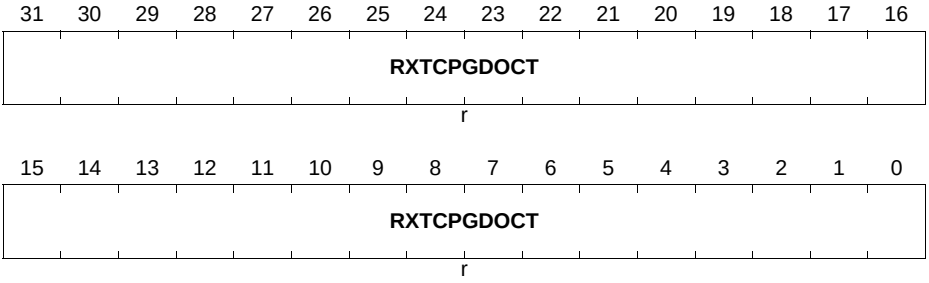


Field	Bits	Type	Description
RXUDPERROCT	[31:0]	r	**RXUDPERROCT** This field indicates the number of bytes received in a UDP segment with checksum errors. This counter does not count the IP Header bytes. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

32-bit Register - RxTCP_Good_Octets

This register maintains the number of bytes received in a good TCP segment.

ETH_RXTCP_GOOD_OCTETS
Register 158 - Receive TCP Good Octet Counter Register (1278_H) Reset Value: $0000\ 0000_H$

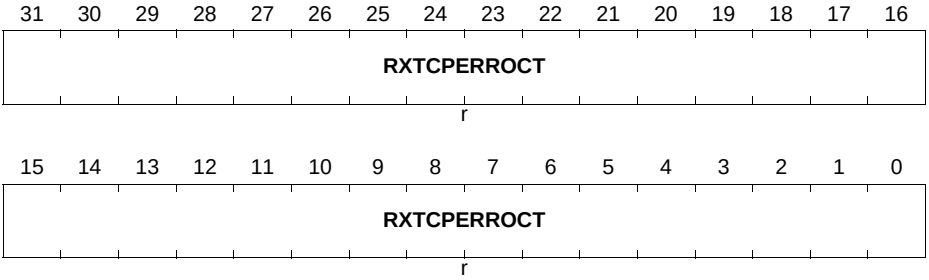


Field	Bits	Type	Description
RXTCPGDOCT	[31:0]	r	**RXTCPGDOCT** This field indicates the number of bytes received in a good TCP segment. This counter does not count the IP Header bytes. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

32-bit Register - RxTCP_Error_Octets

This register maintains the number of bytes received in a TCP segment with checksum errors.

ETH_RXTCP_ERROR_OCTETS
Register 159 - Receive TCP Error Octet Counter Register ($127C_H$) Reset Value: $0000\ 0000_H$

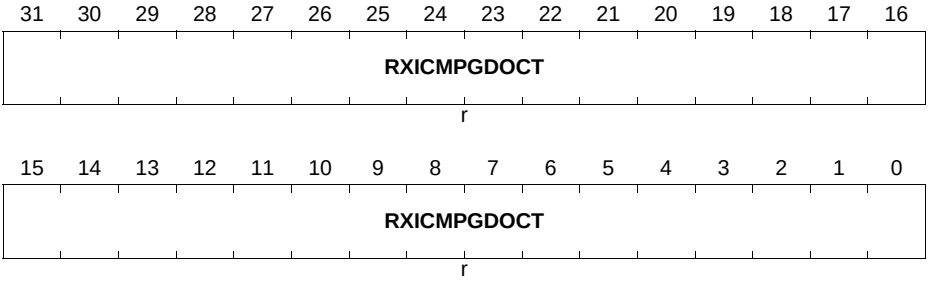


Field	Bits	Type	Description
RXTCPERROCT	[31:0]	r	**RXTCPERROCT** Thsi field indicates the number of bytes received in a TCP segment with checksum errors. This counter does not count the IP Header bytes. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

32-bit Register - RxICMP_Good_Octets

This register maintains the number of bytes received in a good ICMP segment.

ETH_RXICMP_GOOD_OCTETS
Register 160 - Receive ICMP Good Octet Counter Register (1280_H) Reset Value: $0000\ 0000_H$

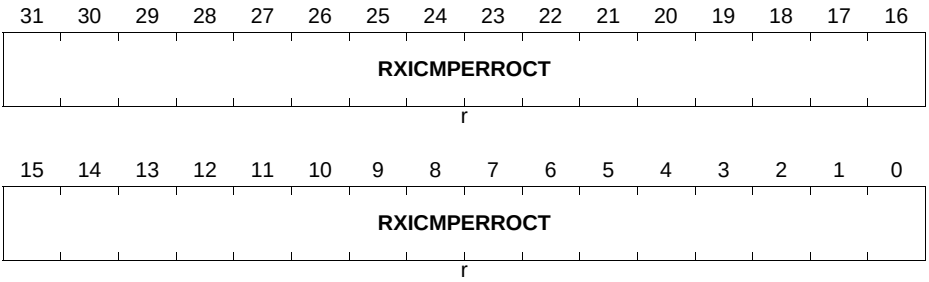


Field	Bits	Type	Description
RXICMPGDOCT	[31:0]	r	**RXICMPGDOCT** This field indicates the number of bytes received in a good ICMP segment. This counter does not count the IP Header bytes. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

32-bit Register - RxICMP_Error_Octets

This register maintains the number of bytes received in a ICMP segment with checksum errors. This counter does not count the IP Header bytes. The Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

ETH_RXICMP_ERROR_OCTETS
Register 161 - Receive ICMP Error Octet Counter Register (1284_H) Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
RXICMPERROCT	[31:0]	r	**RXICMPERROCT** Number of bytes received in an ICMP segment with checksum errors

32-bit Register - Timestamp_Control

This register controls the operation of the System Time generator and the processing of PTP packets for timestamping in the Receiver. Note: * Bits[5:1] are reserved when External Timestamp Input feature is enabled. * Bits[19:8] are reserved and read-only when Advanced Timestamp feature is not enabled. * Bits[28:24] are reserved and read-only when Auxiliary Snapshot feature is not enabled. * Release 3.60a onwards, the functions of Bits 17 and 16 (SNAPTYPSEL) have changed. These functions are not backward compatible with the functions described in release 3.50a.

ETH_TIMESTAMP_CONTROL

Register 448 - Timestamp Control Register (1700_H) **Reset Value: $0000\ 2000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED_31_29			ATSEN3	ATSEN2	ATSEN1	ATSEN0	ATSFC	RESERVED_23_19					TSENMACADDR	SNAPTYPSEL	
r			r	r	r	r	r	r					rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSMSTRENA	TSEVNTENA	TSIPV4ENA	TSIPV6ENA	TSIPENA	TSVER2ENA	TSCTRLSSR	TSENALL	RESERVED_7_6		TSADDREG	TSTRIG	TSUPDT	TSINIT	TSCFUPDT	TSENA
rw	rw	rw	rw	rw	rw	rw	rw	r		rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
TSENA	0	rw	**Timestamp Enable** When set, the timestamp is added for the transmit and receive frames. When disabled, timestamp is not added for the transmit and receive frames and the Timestamp Generator is also suspended. You need to initialize the Timestamp (system time) after enabling this mode. On the receive side, the MAC processes the 1588 frames only if this bit is set.
TSCFUPDT	1	rw	**Timestamp Fine or Coarse Update** When set, this bit indicates that the system times update should be done using the fine update method. When reset, it indicates the system timestamp update should be done using the Coarse method.

Field	Bits	Type	Description
TSINIT	2	rw	**Timestamp Initialize** When set, the system time is initialized (overwritten) with the value specified in the Register 452 (System Time - Seconds Update Register) and Register 453 (System Time - Nanoseconds Update Register). This bit should be read zero before updating it. This bit is reset when the initialization is complete. The Timestamp Higher Word register (if enabled during core configuration) can only be initialized.
TSUPDT	3	rw	**Timestamp Update** When set, the system time is updated (added or subtracted) with the value specified in Register 452 (System Time - Seconds Update Register) and Register 453 (System Time - Nanoseconds Update Register). This bit should be read zero before updating it. This bit is reset when the update is completed in hardware. The Timestamp Higher Word register (if enabled during core configuration) is not updated.
TSTRIG	4	rw	**Timestamp Interrupt Trigger Enable** When set, the timestamp interrupt is generated when the System Time becomes greater than the value written in the Target Time register. This bit is reset after the generation of the Timestamp Trigger Interrupt.
TSADDREG	5	rw	**Addend Reg Update** When set, the content of the Timestamp Addend register is updated in the PTP block for fine correction. This is cleared when the update is completed. This register bit should be zero before setting it.
RESERVED_7_6	[7:6]	r	**RESERVED_7_6**
TSENALL	8	rw	**Enable Timestamp for All Frames** When set, the timestamp snapshot is enabled for all frames received by the MAC.

Field	Bits	Type	Description
TSCTRLSSR	9	rw	**Timestamp Digital or Binary Rollover Control** When set, the Timestamp Low register rolls over after 0x3B9A_C9FF value (that is, 1 nanosecond accuracy) and increments the timestamp (High) seconds. When reset, the rollover value of sub-second register is 0x7FFF_FFFF. The sub-second increment has to be programmed correctly depending on the PTP reference clock frequency and the value of this bit.
TSVER2ENA	10	rw	**Enable PTP packet Processing for Version 2 Format** When set, the PTP packets are processed using the 1588 version 2 format. Otherwise, the PTP packets are processed using the version 1 format.
TSIPENA	11	rw	**Enable Processing of PTP over Ethernet Frames** When set, the MAC receiver processes the PTP packets encapsulated directly in the Ethernet frames. When this bit is clear, the MAC ignores the PTP over Ethernet packets.
TSIPV6ENA	12	rw	**Enable Processing of PTP Frames Sent Over IPv6-UDP** When set, the MAC receiver processes PTP packets encapsulated in UDP over IPv6 packets. When this bit is clear, the MAC ignores the PTP transported over UDP-IPv6 packets.
TSIPV4ENA	13	rw	**Enable Processing of PTP Frames Sent over IPv4-UDP** When set, the MAC receiver processes the PTP packets encapsulated in UDP over IPv4 packets. When this bit is clear, the MAC ignores the PTP transported over UDP-IPv4 packets. This bit is set by default.
TSEVNTENA	14	rw	**Enable Timestamp Snapshot for Event Messages** When set, the timestamp snapshot is taken only for event messages (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When reset, the snapshot is taken for all messages except Announce, Management, and Signaling.
TSMSTRENA	15	rw	**Enable Snapshot for Messages Relevant to Master** When set, the snapshot is taken only for the messages relevant to the master node. Otherwise, the snapshot is taken for the messages relevant to the slave node.

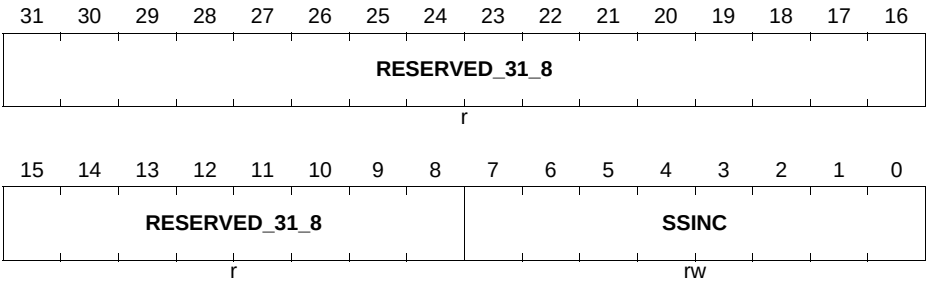
Field	Bits	Type	Description
SNAPTYPSEL	[17:16]	rw	**Select PTP packets for Taking Snapshots** These bits along with Bits 15 and 14 decide the set of PTP packet types for which snapshot needs to be taken.
TSENMACADDR	18	rw	**Enable MAC address for PTP Frame Filtering** When set, the DA MAC address (that matches any MAC Address register) is used to filter the PTP frames when PTP is directly sent over Ethernet.
RESERVED_23_19	[23:19]	r	**RESERVED_23_19**
ATSFC	24	r	**Auxiliary Snapshot FIFO Clear** When set, it resets the pointers of the Auxiliary Snapshot FIFO. This bit is cleared when the pointers are reset and the FIFO is empty. When this bit is high, auxiliary snapshots get stored in the FIFO. This bit is reserved when the Add IEEE 1588 Auxiliary Snapshot option is not selected during core configuration.
ATSEN0	25	r	**Auxiliary Snapshot 0 Enable** This field controls capturing the Auxiliary Snapshot Trigger 0. When this bit is set, the Auxiliary snapshot of event on ptp_aux_trig_i[0] input is enabled. When this bit is reset, the events on this input are ignored. This bit is reserved when the IEEE 1588 Auxiliary Snapshot option is not selected during core configuration.
ATSEN1	26	r	**Auxiliary Snapshot 1 Enable** This field controls capturing the Auxiliary Snapshot Trigger 1. When this bit is set, the Auxiliary snapshot of event on ptp_aux_trig_i[1] input is enabled. When this bit is reset, the events on this input are ignored. This bit is reserved when the IEEE 1588 Auxiliary Snapshot option is not selected during core configuration or the selected number in the Number of IEEE 1588 Auxiliary Snapshot Inputs option is less than two.

Field	Bits	Type	Description
ATSEN2	27	r	**Auxiliary Snapshot 2 Enable** This field controls capturing the Auxiliary Snapshot Trigger 2. When this bit is set, the Auxiliary snapshot of event on ptp_aux_trig_i[2] input is enabled. When this bit is reset, the events on this input are ignored. This bit is reserved when the IEEE 1588 Auxiliary Snapshot option is not selected during core configuration or the selected number in the Number of IEEE 1588 Auxiliary Snapshot Inputs option is less than three.
ATSEN3	28	r	**Auxiliary Snapshot 3 Enable** This field controls capturing the Auxiliary Snapshot Trigger 3. When this bit is set, the Auxiliary snapshot of event on ptp_aux_trig_i[3] input is enabled. When this bit is reset, the events on this input are ignored. This bit is reserved when the IEEE 1588 Auxiliary Snapshot option is not selected during core configuration or the selected number in the Number of IEEE 1588 Auxiliary Snapshot Inputs option is less than four.
RESERVED_31_29	[31:29]	r	**RESERVED_31_29**

32-bit Register - Sub_Second_Increment

This register is present only when the IEEE 1588 timestamp feature is selected without an external timestamp input. In the Coarse Update mode (TSCFUPDT bit in Register 448), the value in this register is added to the system time every clock cycle of clk_ptp_ref_i. In the Fine Update mode, the value in this register is added to the system time whenever the Accumulator gets an overflow.

ETH_SUB_SECOND_INCREMENT
Register 449 - Sub-Second Increment Register (1704_H) Reset Value: $0000\ 0000_H$

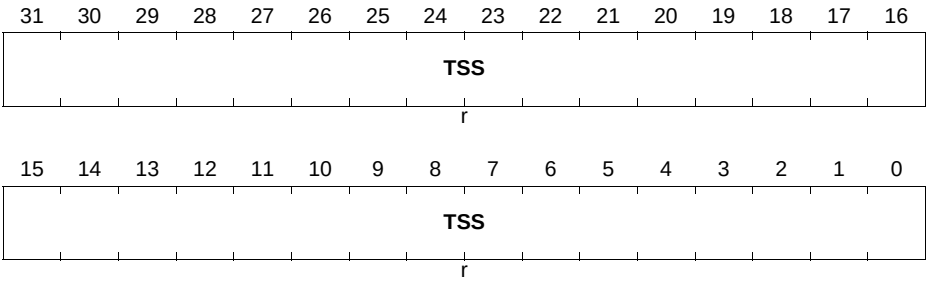


Field	Bits	Type	Description
SSINC	[7:0]	rw	**Sub-second Increment Value** The value programmed in this field is accumulated every clock cycle (of clk_ptp_i) with the contents of the sub-second register. For example, when PTP clock is 50 MHz (period is 20 ns), you should program 20 (0x14) when the System Time-Nanoseconds register has an accuracy of 1 ns (TSCTRLSSR bit is set). When TSCTRLSSR is clear, the Nanoseconds register has a resolution of ~0.465ns. In this case, you should program a value of 43 (0x2B) that is derived by 20ns/0.465.
RESERVED_31_8	[31:8]	r	**RESERVED_31_8**

32-bit Register - System_Time_Seconds

The System Time -Seconds register, along with System-TimeNanoseconds register, indicates the current value of the system time maintained by the MAC. Though it is updated on a continuous basis, there is some delay from the actual time because of clock domain transfer latencies (from clk_ptp_ref_i to clk_csr_i). These registers (450 and 451) are present only when the IEEE 1588 Timestamp feature is selected without external timestamp input.

ETH_SYSTEM_TIME_SECONDS
Register 450 - System Time - Seconds Register (1708_H) Reset Value: $0000\ 0000_H$

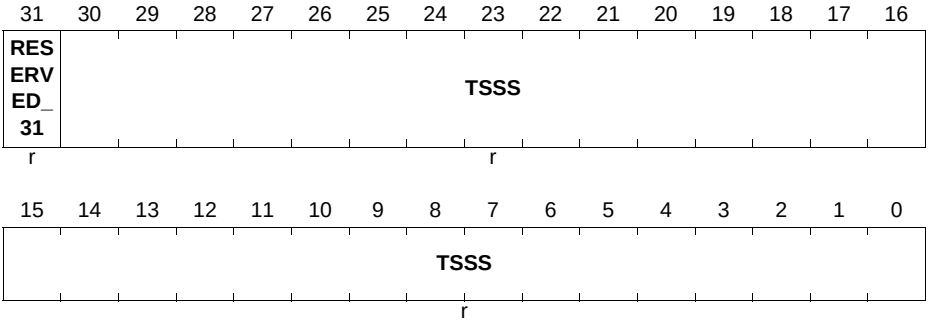


Field	Bits	Type	Description
TSS	[31:0]	r	**Timestamp Second** The value in this field indicates the current value in seconds of the System Time maintained by the MAC.

32-bit Register - System_Time_Nanoseconds

The value in this field has the sub second representation of time, with an accuracy of 0.46 ns. When TSCTRLSSR is set, each bit represents 1 ns and the maximum value is 0x3B9A_C9FF, after which it rolls-over to zero.

ETH_SYSTEM_TIME_NANOSECONDS
Register 451 - System Time - Nanoseconds Register ($170C_H$) Reset Value: 0000 0000_H

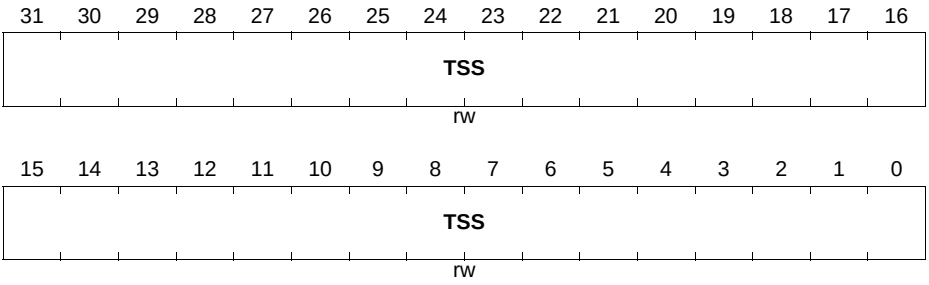


Field	Bits	Type	Description
TSSS	[30:0]	r	**Timestamp Sub Seconds** The value in this field has the sub second representation of time, with an accuracy of 0.46 ns. When bit 9 (TSCTRLSSR) is set in Register 448 (Timestamp Control Register), each bit represents 1 ns and the maximum value is 0x3B9A_C9FF, after which it rolls-over to zero.
RESERVED_31	31	r	**RESERVED_31**

32-bit Register - System_Time_Seconds_Update

The System Time - Seconds Update register, along with the System Time - Nanoseconds Update register, initializes or updates the system time maintained by the MAC. You must write both of these registers before setting the TSINIT or TSUPDT bits in the Timestamp Control register. This register is present only when the IEEE 1588 Timestamp feature is selected without external timestamp input.

ETH_SYSTEM_TIME_SECONDS_UPDATE
Register 452 - System Time - Seconds Update Register (1710_H) Reset Value: 0000 0000_H

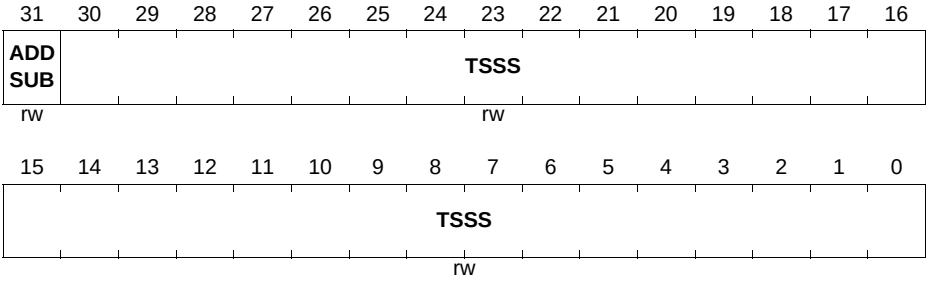


Field	Bits	Type	Description
TSS	[31:0]	rw	**Timestamp Second** The value in this field indicates the time in seconds to be initialized or added to the system time.

32-bit Register - System_Time_Nanoseconds_Update

This register is present only when IEEE 1588 timestamp feature is selected without external timestamp input.

ETH_SYSTEM_TIME_NANOSECONDS_UPDATE
Register 453 - System Time - Nanoseconds Update Register (1714_H) Reset Value: $0000\ 0000_H$

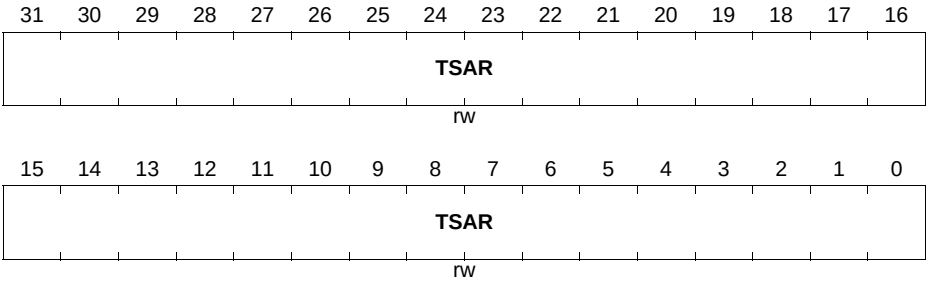


Field	Bits	Type	Description
TSSS	[30:0]	rw	**Timestamp Sub Second** The value in this field has the sub second representation of time, with an accuracy of 0.46 ns. When bit 9 (TSCTRLSSR) is set in Register 448 (Timestamp Control Register), each bit represents 1 ns and the programmed value should not exceed 0x3B9A_C9FF.
ADDSUB	31	rw	**Add or subtract time** When this bit is set, the time value is subtracted with the contents of the update register. When this bit is reset, the time value is added with the contents of the update register.

32-bit Register - Timestamp_Addend

This register is present only when the IEEE 1588 Timestamp feature is selected without external timestamp input. This register value is used only when the system time is configured for Fine Update mode (TSCFUPDT bit in Register 448). This register content is added to a 32-bit accumulator in every clock cycle (of clk_ptp_ref_i) and the system time is updated whenever the accumulator overflows.

ETH_TIMESTAMP_ADDEND
Register 454 - Timestamp Addend Register (1718_H) **Reset Value: $0000\ 0000_H$**

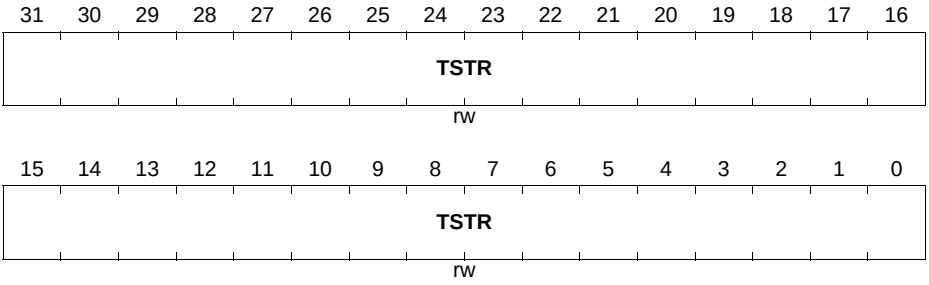


Field	Bits	Type	Description
TSAR	[31:0]	rw	**Timestamp Addend Register** This field indicates the 32-bit time value to be added to the Accumulator register to achieve time synchronization.

32-bit Register - Target_Time_Seconds

The Target Time Seconds register, along with Target Time Nanoseconds register, is used to schedule an interrupt event (Register 458[1] when Advanced Timestamping is enabled; otherwise, TS interrupt bit in Register14[9]) when the system time exceeds the value programmed in these registers. This register is present only when the IEEE 1588 Timestamp feature is selected without external timestamp input.

ETH_TARGET_TIME_SECONDS
Register 455 - Target Time Seconds Register ($171C_H$) **Reset Value: $0000\ 0000_H$**

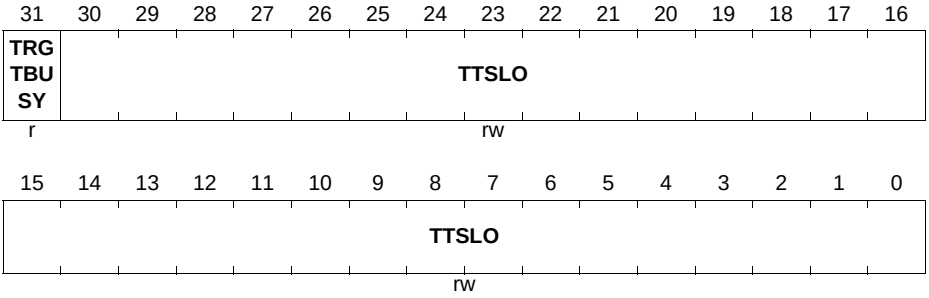


Field	Bits	Type	Description
TSTR	[31:0]	rw	**Target Time Seconds Register** This register stores the time in seconds. When the timestamp value matches or exceeds both Target Timestamp registers, then based on Bits [6:5] of Register 459 (PPS Control Register), the MAC starts or stops the PPS signal output and generates an interrupt (if enabled).

32-bit Register - Target_Time_Nanoseconds

This register is present only when the IEEE 1588 Timestamp feature is selected without external timestamp input.

ETH_TARGET_TIME_NANOSECONDS
Register 456 - Target Time Nanoseconds Register (1720_H) Reset Value: $0000\ 0000_H$



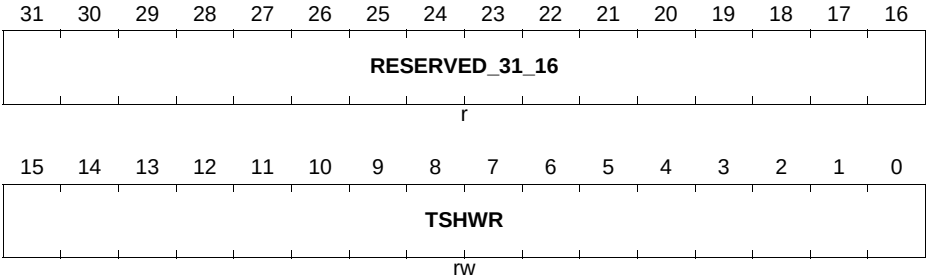
Field	Bits	Type	Description
TTSLO	[30:0]	rw	**Target Timestamp Low Register** This register stores the time in (signed) nanoseconds. When the value of the timestamp matches the both Target Timestamp registers, then based on the TRGTMODSEL0 field (Bits [6:5]) in Register 459 (PPS Control Register), the MAC starts or stops the PPS signal output and generates an interrupt (if enabled). This value should not exceed 0x3B9A_C9FF when TSCTRLSSR is set in the Timestamp control register. The actual start or stop time of the PPS signal output may have an error margin up to one unit of sub-second increment value.

Field	Bits	Type	Description
TRGTBUSY	31	r	**Target Time Register Busy** The MAC sets this bit when the PPSCMD field (Bits[3:0]) in Register 459 (PPS Control Register) is programmed to 010 or 011. Programming the PPSCMD field to 010 or 011, instructs the MAC to synchronize the Target Time Registers to the PTP clock domain. The MAC clears this bit after synchronizing the Target Time Registers to the PTP clock domain The application must not update the Target Time Registers when this bit is read as 1. Otherwise, the synchronization of the previous programmed time gets corrupted. This bit is reserved when the Enable Flexible Pulse-Per-Second Output feature is not selected.

32-bit Register - System_Time_Higher_Word_Seconds

This register is present only when the IEEE 1588 Advanced Timestamp feature is selected without an external timestamp input.

ETH_SYSTEM_TIME_HIGHER_WORD_SECONDS
Register 457 - System Time - Higher Word Seconds Register (1724_H) Reset Value: $0000\ 0000_H$

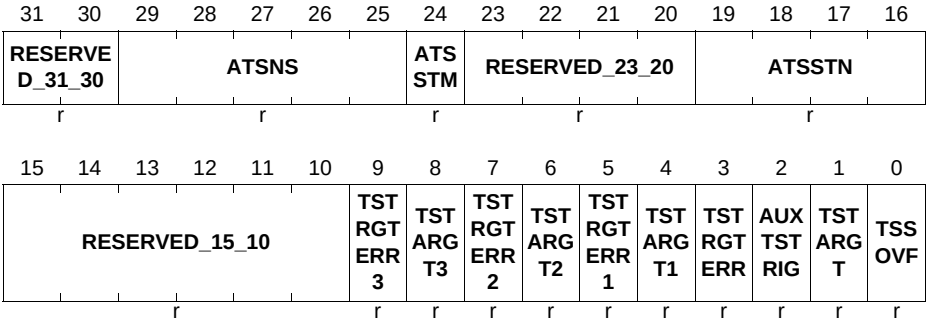


Field	Bits	Type	Description
TSHWR	[15:0]	rw	**Timestamp Higher Word Register** This field contains the most significant 16-bits of the timestamp seconds value. This register is optional and can be selected using the Enable IEEE 1588 Higher Word Register option during core configuration. The register is directly written to initialize the value. This register is incremented when there is an overflow from the 32-bits of the System Time - Seconds register.
RESERVED_31_16	[31:16]	r	**RESERVED_31_16**

32-bit Register - Timestamp_Status

This register is present only when Advanced IEEE 1588 Timestamp feature is selected. All bits except Bits[27:25] gets cleared when the host reads this register.

ETH_TIMESTAMP_STATUS
Register 458 - Timestamp Status Register (1728_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
TSSOVF	0	r	**Timestamp Seconds Overflow** When set, this bit indicates that the seconds value of the timestamp (when supporting version 2 format) has overflowed beyond 32'hFFFF_FFFF.
TSTARGT	1	r	**Timestamp Target Time Reached** When set, this bit indicates that the value of system time is greater or equal to the value specified in the Register 455 (Target Time Seconds Register) and Register 456 (Target Time Nanoseconds Register).
AUXTSTRIG	2	r	**Auxiliary Timestamp Trigger Snapshot** This bit is set high when the auxiliary snapshot is written to the FIFO. This bit is valid only if the Enable IEEE 1588 Auxiliary Snapshot feature is selected.
TSTRGTERR	3	r	**Timestamp Target Time Error** This bit is set when the target time, being programmed in Target Time Registers, is already elapsed. This bit is cleared when read by the application.

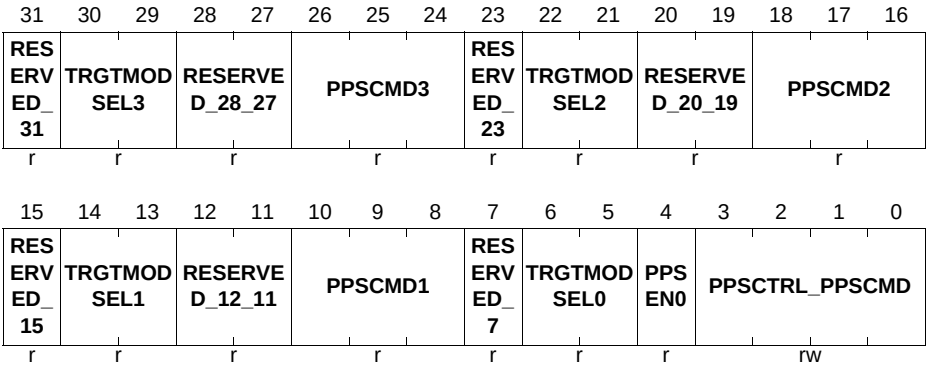
Field	Bits	Type	Description
TSTARGT1	4	r	**Timestamp Target Time Reached for Target Time PPS1** When set, this bit indicates that the value of system time is greater than or equal to the value specified in Register 480 (PPS1 Target Time High Register) and Register 481 (PPS1 Target Time Low Register).
TSTRGTERR1	5	r	**Timestamp Target Time Error** This bit is set when the target time, being programmed in Register 480 and Register 481, is already elapsed. This bit is cleared when read by the application.
TSTARGT2	6	r	**Timestamp Target Time Reached for Target Time PPS2** When set, this bit indicates that the value of system time is greater than or equal to the value specified in Register 488 (PPS2 Target Time High Register) and Register 489 (PPS2 Target Time Low Register).
TSTRGTERR2	7	r	**Timestamp Target Time Error** This bit is set when the target time, being programmed in Register 488 and Register 489, is already elapsed. This bit is cleared when read by the application.
TSTARGT3	8	r	**Timestamp Target Time Reached for Target Time PPS3** When set, this bit indicates that the value of system time is greater than or equal to the value specified in Register 496 (PPS3 Target Time High Register) and Register 497 (PPS3 Target Time Low Register).
TSTRGTERR3	9	r	**Timestamp Target Time Error** This bit is set when the target time, being programmed in Register 496 and Register 497, is already elapsed. This bit is cleared when read by the application.
RESERVED_15_10	[15:10]	r	**RESERVED_15_10**

Field	Bits	Type	Description
ATSSTN	[19:16]	r	**Auxiliary Timestamp Snapshot Trigger Identifier** These bits identify the Auxiliary trigger inputs for which the timestamp available in the Auxiliary Snapshot Register is applicable. When more than one bit is set at the same time, it means that corresponding auxiliary triggers were sampled at the same clock. These bits are applicable only if the number of Auxiliary snapshots is more than one. One bit is assigned for each trigger as shown in the following list: * Bit 16: Auxiliary trigger 0 * Bit 17: Auxiliary trigger 1 * Bit 18: Auxiliary trigger 2 * Bit 19: Auxiliary trigger 3 The software can read this register to find the triggers that are set when the timestamp is taken.
RESERVED_23_20	[23:20]	r	**RESERVED_23_20**
ATSSTM	24	r	**Auxiliary Timestamp Snapshot Trigger Missed** This bit is set when the Auxiliary timestamp snapshot FIFO is full and external trigger was set. This indicates that the latest snapshot is not stored in the FIFO. This bit is valid only if the Add IEEE 1588 Auxiliary Snapshot option is selected during core configuration.
ATSNS	[29:25]	r	**Number of Auxiliary Timestamp Snapshots** This field indicates the number of Snapshots available in the FIFO. A value equal to the selected depth of FIFO (4, 8, or 16) indicates that the Auxiliary Snapshot FIFO is full. These bits are cleared (to 00000) when the Auxiliary snapshot FIFO clear bit is set. This bit is valid only if the Add IEEE 1588 Auxiliary Snapshot option is selected during core configuration.
RESERVED_31_30	[31:30]	r	**RESERVED_31_30**

32-bit Register - PPS_Control

This register is present only when the Advanced Timestamp feature is selected and External Timestamp is not enabled. Note: * Bits[30:24] are valid only when four Flexible PPS outputs are selected. * Bits[22:16] are valid only when three or more Flexible PPS outputs are selected. * Bits[14:8] are valid only when two or more Flexible PPS outputs are selected. * Bits[6:4] are valid only when Flexible PPS feature is selected.

ETH_PPS_CONTROL
Register 459 - PPS Control Register (172C$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
PPSCTRL_PPSCMD	[3:0]	rw	**PPSCTRL0 or PPSCMD0** PPSCTRL0: PPS0 Output Frequency Control This field controls the frequency of the PPS0 output (ptp_pps_o[0]) signal. The default value of PPSCTRL is 0000, and the PPS output is 1 pulse (of width clk_ptp_i) every second. For other values of PPSCTRL, the PPS output becomes a generated clock of following frequencies: -0001: The binary rollover is 2 Hz, and the digital rollover is 1 Hz. -0010: The binary rollover is 4 Hz, and the digital rollover is 2 Hz. -0011: The binary rollover is 8 Hz, and the digital rollover is 4 Hz. -0100: The binary rollover is 16 Hz, and the digital rollover is 8 Hz. -... -1111: The binary rollover is 32.768 KHz, and the digital rollover is 16.384 KHz. Note: In the binary rollover mode, the PPS output (ptp_pps_o) has a duty cycle of 50 percent with these frequencies. In the digital rollover mode, the PPS output frequency is an average number. The actual clock is of different frequency that gets synchronized every second. For example: * When PPSCTRL = 0001, the PPS (1 Hz) has a low period of 537 ms and a high period of 463 ms * When PPSCTRL = 0010, the PPS (2 Hz) is a sequence of: - One clock of 50 percent duty cycle and 537 ms period - Second clock of 463 ms period (268 ms low and 195 ms high) * When PPSCTRL = 0011, the PPS (4 Hz) is a sequence of: - Three clocks of 50 percent duty cycle and 268 ms period - Fourth clock of 195 ms period (134 ms low and 61 ms high) This behavior is because of the non-linear toggling of bits in the digital rollover mode in Register 451 (System Time - Nanoseconds Register). Flexible PPS0 Output (ptp_pps_o[0]) Control Programming these bits with a non-zero value instructs the MAC to initiate an event. Once the command is

Field	Bits	Type	Description
PPSEN0	4	r	**Flexible PPS Output Mode Enable** When set low, Bits[3:0] function as PPSCTRL (backward compatible). When set high, Bits[3:0] function as PPSCMD.
TRGTMODSEL0	[6:5]	r	**Target Time Register Mode for PPS0 Output** This field indicates the Target Time registers (register 455 and 456) mode for PPS0 output signal: * 00: Indicates that the Target Time registers are programmed only for generating the interrupt event. * 01: Reserved * 10: Indicates that the Target Time registers are programmed for generating the interrupt event and starting or stopping the generation of the PPS0 output signal. * 11: Indicates that the Target Time registers are programmed only for starting or stopping the generation of the PPS0 output signal. No interrupt is asserted.
RESERVED_7	7	r	**RESERVED_7**
PPSCMD1	[10:8]	r	**Flexible PPS1 Output Control** This field controls the flexible PPS1 output (ptp_pps_o[1]) signal. This field is similar to PPSCMD0[2:0] in functionality.
RESERVED_12_11	[12:11]	r	**RESERVED_12_11**
TRGTMODSEL1	[14:13]	r	**Target Time Register Mode for PPS1 Output** This field indicates the Target Time registers (register 480 and 481) mode for PPS1 output signal. This field is similar to the TRGTMODSEL0 field.
RESERVED_15	15	r	**RESERVED_15**
PPSCMD2	[18:16]	r	**Flexible PPS2 Output Control** This field controls the flexible PPS2 output (ptp_pps_o[2]) signal. This field is similar to PPSCMD0[2:0] in functionality.
RESERVED_20_19	[20:19]	r	**RESERVED_20_19**

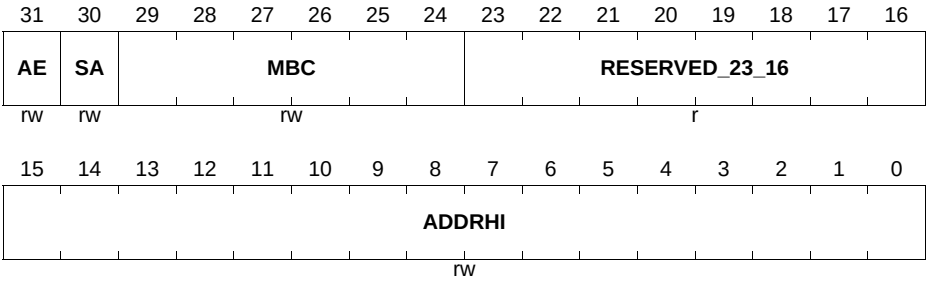
Field	Bits	Type	Description
TRGTMODSEL2	[22:21]	r	**Target Time Register Mode for PPS2 Output** This field indicates the Target Time registers (register 488 and 489) mode for PPS2 output signal. This field is similar to the TRGTMODSEL0 field.
RESERVED_23	23	r	**RESERVED_23**
PPSCMD3	[26:24]	r	**Flexible PPS3 Output Control** This field controls the flexible PPS3 output (ptp_pps_o[3]) signal. This field is similar to PPSCMD0[2:0] in functionality.
RESERVED_28_27	[28:27]	r	**RESERVED_28_27**
TRGTMODSEL3	[30:29]	r	**Target Time Register Mode for PPS3 Output** This field indicates the Target Time registers (register 496 and 497) mode for PPS3 output signal. This field is similar to the TRGTMODSEL0 field.
RESERVED_31	31	r	**RESERVED_31**

32-bit Register - MAC_Address16_High

The MAC Address16 High register holds the upper 16 bits of the 17th 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address16 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address16 Low Register must be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS16_HIGH

Register 512 - MAC Address16 High Register (1800_H) **Reset Value: $0000\ FFFF_H$**



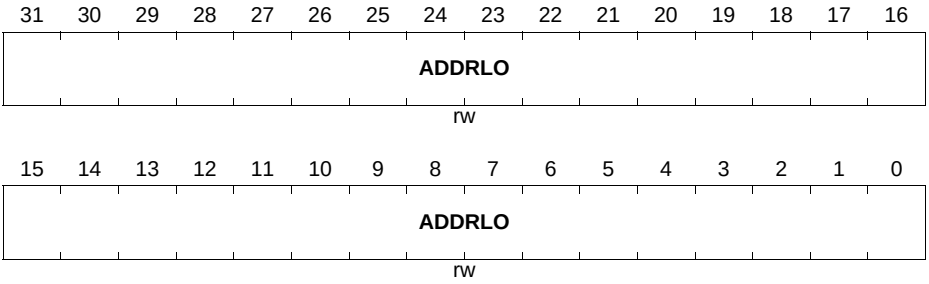
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address16 [47:32]** This field contains the upper 16 bits (47:32) of the 17th 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address16 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address16[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address16[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the 17th MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address16_Low

The MAC Address16 Low register holds the lower 32 bits of the 17th 6-byte MAC address of the station.

ETH_MAC_ADDRESS16_LOW
Register 513 - MAC Address16 Low Register (1804_H) **Reset Value: $FFFF\ FFFF_H$**



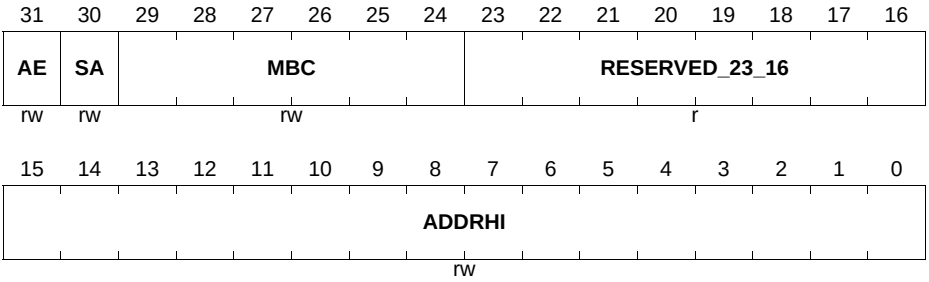
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address16 [31:0]** This field contains the lower 32 bits of the 17th 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address17_High

The MAC Address17 High register holds the upper 16 bits of the 18th 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address17 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address17 Low Register must be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS17_HIGH

Register 514 - MAC Address17 High Register (1808_H) **Reset Value: $0000\ FFFF_H$**



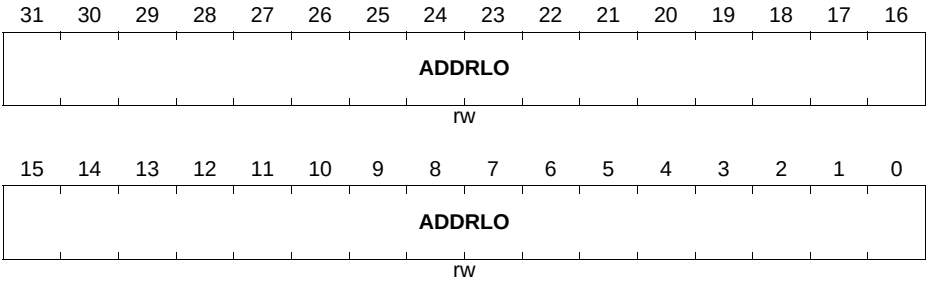
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address18 [47:32]** This field contains the upper 16 bits (47:32) of the 19th 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address17 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address17[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address17[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the 18th MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address17_Low

The MAC Address17 Low register holds the lower 32 bits of the 18th 6-byte MAC address of the station.

ETH_MAC_ADDRESS17_LOW
Register 515 - MAC Address17 Low Register (180C$_H$) Reset Value: FFFF FFFF$_H$



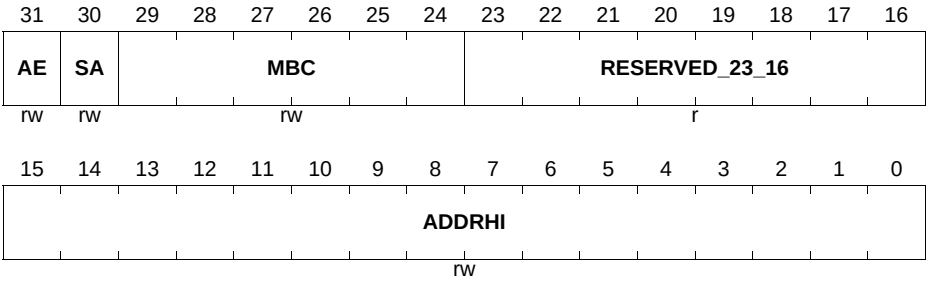
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address17 [31:0]** This field contains the lower 32 bits of the 18th 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address18_High

The MAC Address18 High register holds the upper 16 bits of the 19th 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address18 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address18 Low Register must be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS18_HIGH

Register 516 - MAC Address18 High Register (1810_H) **Reset Value: $0000\ FFFF_H$**



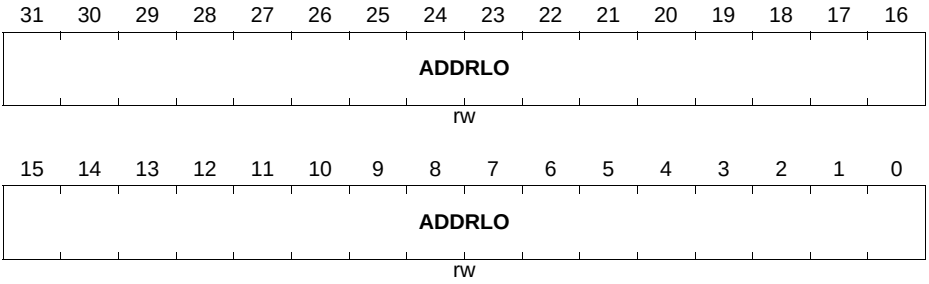
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address18 [47:32]** This field contains the upper 16 bits (47:32) of the 19th 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address18 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address18[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address18[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the 19th MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address18_Low

The MAC Address18 Low register holds the lower 32 bits of the 19th 6-byte MAC address of the station.

ETH_MAC_ADDRESS18_LOW
Register 517 - MAC Address18 Low Register (1814_H) **Reset Value: FFFF FFFF$_H$**



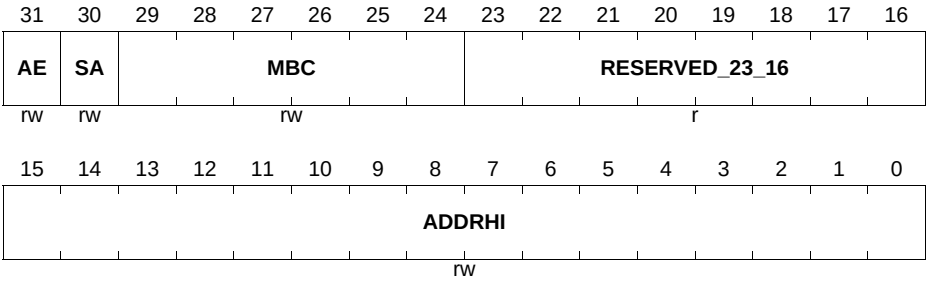
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address18 [31:0]** This field contains the lower 32 bits of the 19th 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address19_High

The MAC Address19 High register holds the upper 16 bits of the 20th 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address19 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address19 Low Register must be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS19_HIGH

Register 518 - MAC Address19 High Register (1818_H) **Reset Value: $0000\ FFFF_H$**



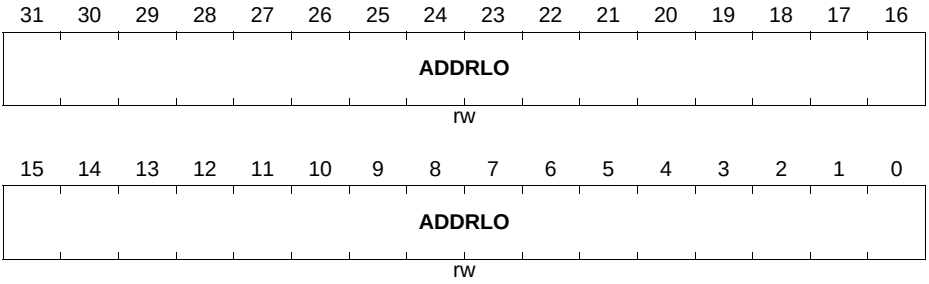
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address19 [47:32]** This field contains the upper 16 bits (47:32) of the 20th 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address19 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address19[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address19[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the 20th MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address19_Low

The MAC Address19 Low register holds the lower 32 bits of the 20th 6-byte MAC address of the station.

ETH_MAC_ADDRESS19_LOW
Register 519 - MAC Address19 Low Register ($181C_H$) Reset Value: $FFFF\ FFFF_H$



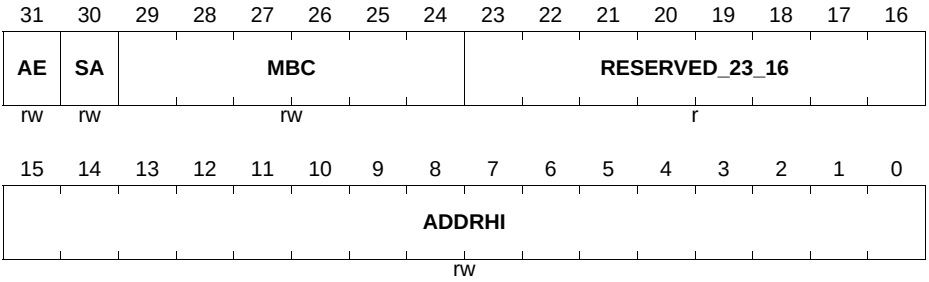
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address19 [31:0]** This field contains the lower 32 bits of the 20th 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address20_High

The MAC Address20 High register holds the upper 16 bits of the 21st 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address20 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address20 Low Register must be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS20_HIGH

Register 520 - MAC Address20 High Register (1820_H) **Reset Value: $0000\ FFFF_H$**



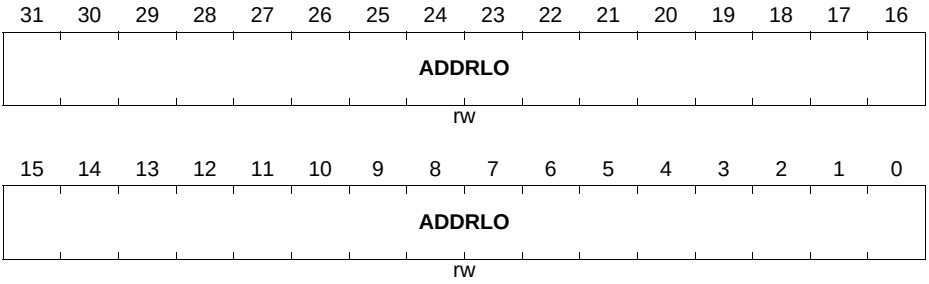
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address20 [47:32]** This field contains the upper 16 bits (47:32) of the 20th 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address20 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address20[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address20[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the 21st MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address20_Low

The MAC Address20 Low register holds the lower 32 bits of the 21st 6-byte MAC address of the station.

ETH_MAC_ADDRESS20_LOW
Register 521 - MAC Address20 Low Register (1824_H) **Reset Value: $FFFF\ FFFF_H$**



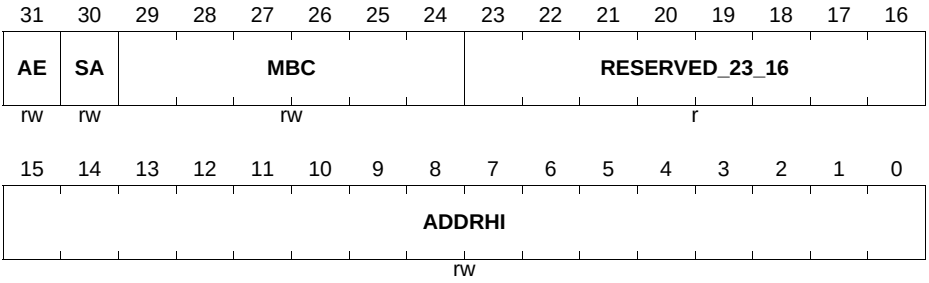
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address20 [31:0]** This field contains the lower 32 bits of the 21st 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address21_High

The MAC Address21 High register holds the upper 16 bits of the 22nd 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address21 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address21 Low Register must be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS21_HIGH

Register 522 - MAC Address21 High Register (1828_H) **Reset Value: $0000\ FFFF_H$**



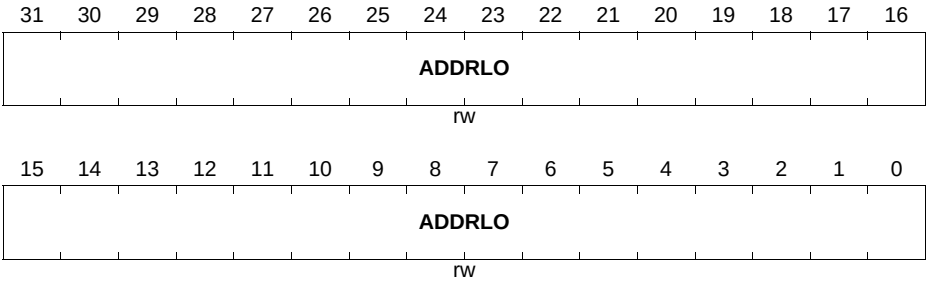
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address21 [47:32]** This field contains the upper 16 bits (47:32) of the 6-byte 22nd MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address21 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address21[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address21[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the 22nd MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address21_Low

The MAC Address21 Low register holds the lower 32 bits of the 22nd 6-byte MAC address of the station.

ETH_MAC_ADDRESS21_LOW
Register 523 - MAC Address21 Low Register ($182C_H$) Reset Value: $FFFF\ FFFF_H$



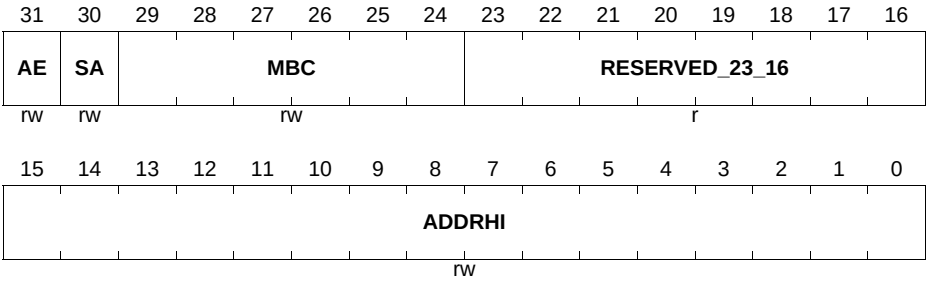
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address21 [31:0]** This field contains the lower 32 bits of the 22nd 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address22_High

The MAC Address22 High register holds the upper 16 bits of the 23rd 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address22 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address22 Low Register must be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS22_HIGH

Register 524 - MAC Address22 High Register (1830_H) **Reset Value: $0000\ FFFF_H$**



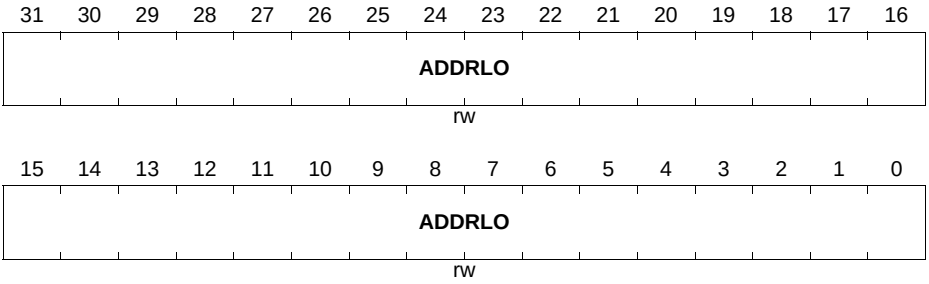
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address22 [47:32]** This field contains the upper 16 bits (47:32) of the 23rd 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address22 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address22[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address22[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the 23rd MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address22_Low

The MAC Address22 Low register holds the lower 32 bits of the 23rd 6-byte MAC address of the station.

ETH_MAC_ADDRESS22_LOW
Register 525 - MAC Address22 Low Register (1834_H) **Reset Value: $FFFF\ FFFF_H$**



Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address22 [31:0]** This field contains the lower 32 bits of the 23rd 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address23_High

The MAC Address23 High register holds the upper 16 bits of the 24th 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address23 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address23 Low Register must be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS23_HIGH
Register 526 - (1838_H) Reset Value: $0000\ FFFF_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	SA	MBC						RESERVED_23_16							
rw	rw	rw						r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRHI															
rw															

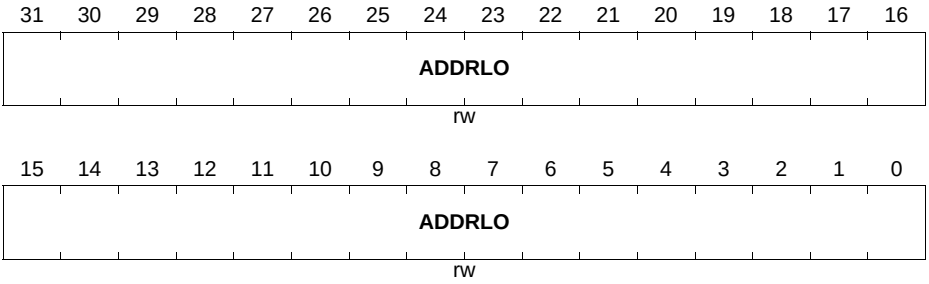
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address23 [47:32]** This field contains the upper 16 bits (47:32) of the 24th 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address23 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address23[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address23[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the 24th MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address23_Low

The MAC Address23 Low register holds the lower 32 bits of the 24th 6-byte MAC address of the station.

ETH_MAC_ADDRESS23_LOW
Register 527 - MAC Address23 Low Register ($183C_H$) Reset Value: $FFFF\ FFFF_H$



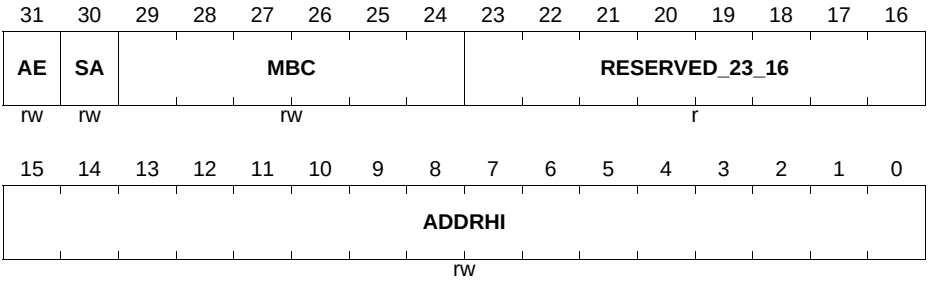
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address23 [31:0]** This field contains the lower 32 bits of the 24th 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address24_High

The MAC Address24 High register holds the upper 16 bits of the 25th 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address24 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address24 Low Register must be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS24_HIGH

Register 528 - MAC Address24 High Register (1840_H) **Reset Value: $0000\ FFFF_H$**



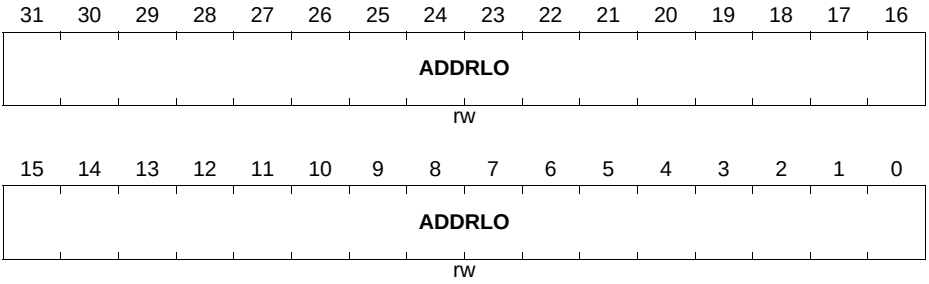
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address1 [47:32]** This field contains the upper 16 bits (47:32) of the 25th 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address24 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address24[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address24[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the 25th MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address24_Low

The MAC Address24 Low register holds the lower 32 bits of the 25th 6-byte MAC address of the station.

ETH_MAC_ADDRESS24_LOW
Register 529 - MAC Address24 Low Register (1844_H) **Reset Value: $FFFF\ FFFF_H$**



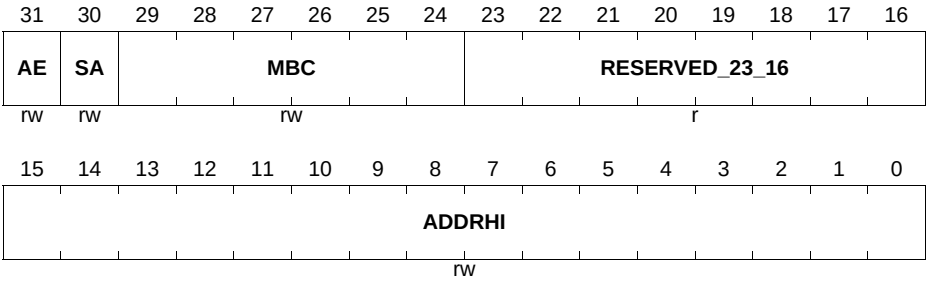
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address24 [31:0]** This field contains the lower 32 bits of the 25th 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address25_High

The MAC Address25 High register holds the upper 16 bits of the 6-byte 26th MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address25 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address25 Low Register must be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS25_HIGH

Register 530 - MAC Address25 High Register (1848_H) **Reset Value: $0000\ FFFF_H$**



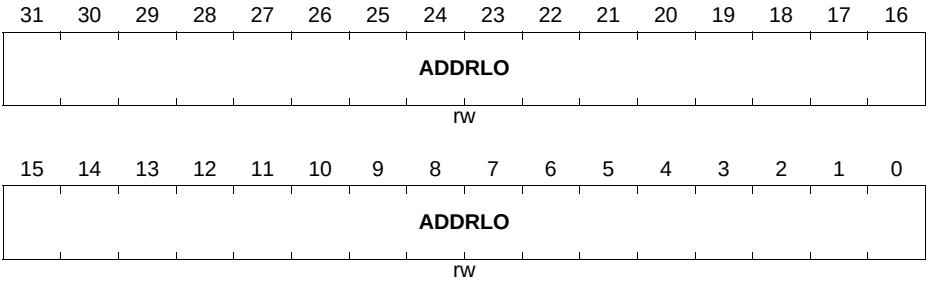
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address25 [47:32]** This field contains the upper 16 bits (47:32) of the 26th 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address25 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address25[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address25[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the 26th MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address25_Low

The MAC Address25 Low register holds the lower 32 bits of the 26th 6-byte MAC address of the station.

ETH_MAC_ADDRESS25_LOW
Register 531 - MAC Address25 Low Register ($184C_H$) Reset Value: $FFFF\ FFFF_H$



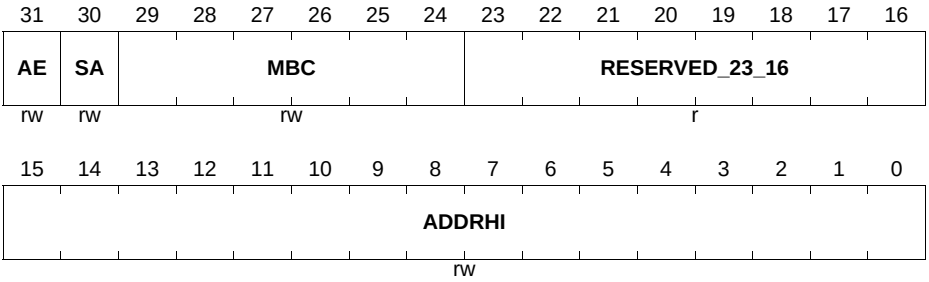
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address25 [31:0]** This field contains the lower 32 bits of the 26th 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address26_High

The MAC Address26 High register holds the upper 16 bits of the 27th 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address26 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address26 Low Register must be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS26_HIGH

Register 532 - MAC Address26 High Register (1850_H) **Reset Value: $0000\ FFFF_H$**



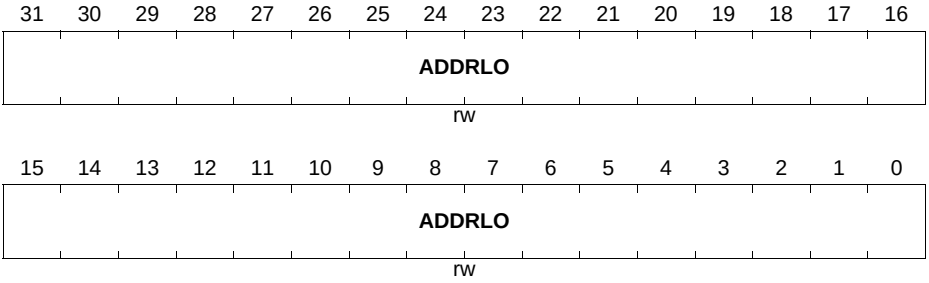
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address26 [47:32]** This field contains the upper 16 bits (47:32) of the 27th 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address26 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address26[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address26[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the 27th MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address26_Low

The MAC Address26 Low register holds the lower 32 bits of the 27th 6-byte MAC address of the station.

ETH_MAC_ADDRESS26_LOW
Register 533 - MAC Address26 Low Register (1854_H) **Reset Value: $FFFF\ FFFF_H$**



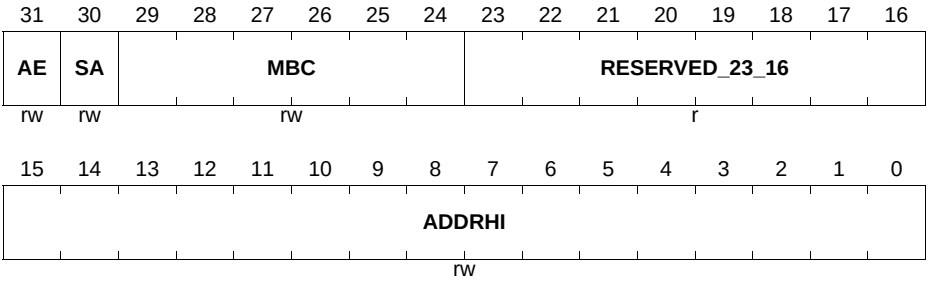
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address26 [31:0]** This field contains the lower 32 bits of the 27th 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address27_High

The MAC Address27 High register holds the upper 16 bits of the 28th 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address27 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address27 Low Register must be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS27_HIGH

Register 534 - MAC Address27 High Register (1858_H) **Reset Value: $0000\ FFFF_H$**



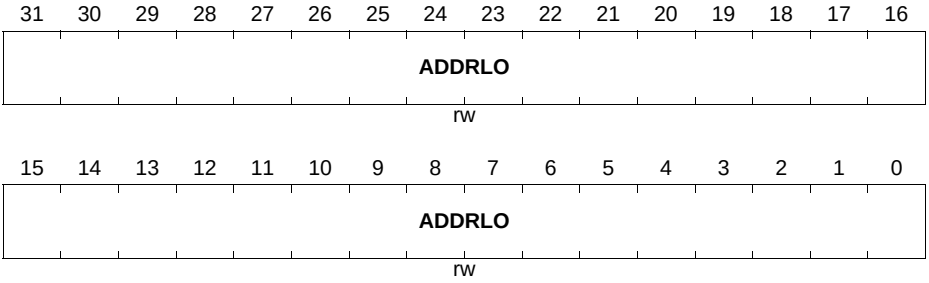
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address27 [47:32]** This field contains the upper 16 bits (47:32) of the 28th 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address27 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address27[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address27[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the 28th MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address27_Low

The MAC Address27 Low register holds the lower 32 bits of the 28th 6-byte MAC address of the station.

ETH_MAC_ADDRESS27_LOW
Register 535 - MAC Address27 Low Register (185C$_H$) Reset Value: FFFF FFFF$_H$



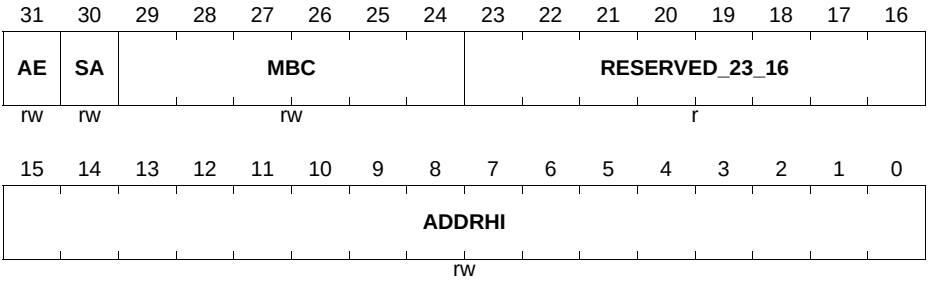
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address27 [31:0]** This field contains the lower 32 bits of the 28th 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address28_High

The MAC Address28 High register holds the upper 16 bits of the 29th 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address28 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address28 Low Register must be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS28_HIGH

Register 536 - MAC Address28 High Register (1860_H) **Reset Value: $0000\ FFFF_H$**



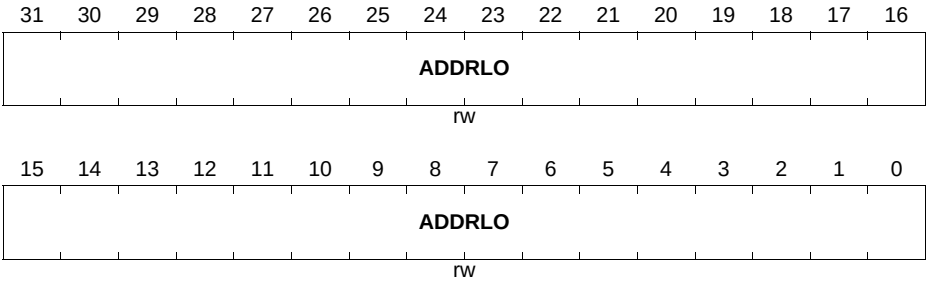
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address28 [47:32]** This field contains the upper 16 bits (47:32) of the 29th 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address28 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address28[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address28[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the 29th MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address28_Low

The MAC Address28 Low register holds the lower 32 bits of the 29th 6-byte MAC address of the station.

ETH_MAC_ADDRESS28_LOW
Register 537 - MAC Address28 Low Register (1864_H) **Reset Value: $FFFF\ FFFF_H$**



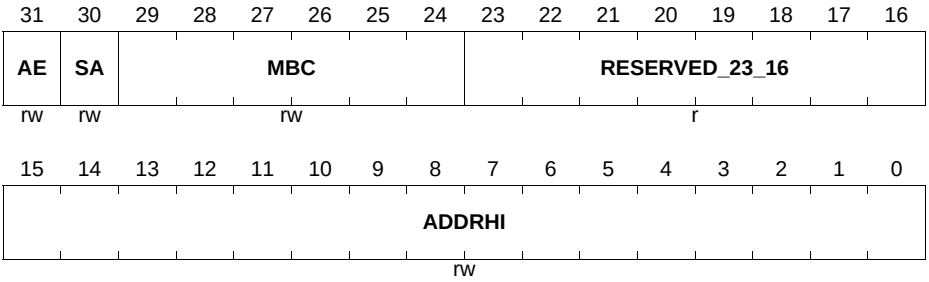
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address28 [31:0]** This field contains the lower 32 bits of the 29th 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address29_High

The MAC Address29 High register holds the upper 16 bits of the 6-byte 30th MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address29 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address29 Low Register must be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS29_HIGH

Register 538 - MAC Address29 High Register (1868_H) **Reset Value: $0000\ FFFF_H$**



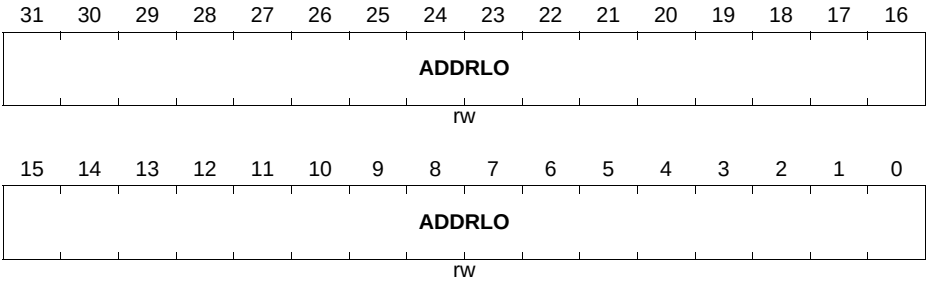
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address29 [47:32]** This field contains the upper 16 bits (47:32) of the 30th 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address29 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address29[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address29[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the 30th MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address29_Low

The MAC Address29 Low register holds the lower 32 bits of the 30th 6-byte MAC address of the station.

ETH_MAC_ADDRESS29_LOW
Register 539 - MAC Address29 Low Register (186C$_H$) Reset Value: FFFF FFFF$_H$



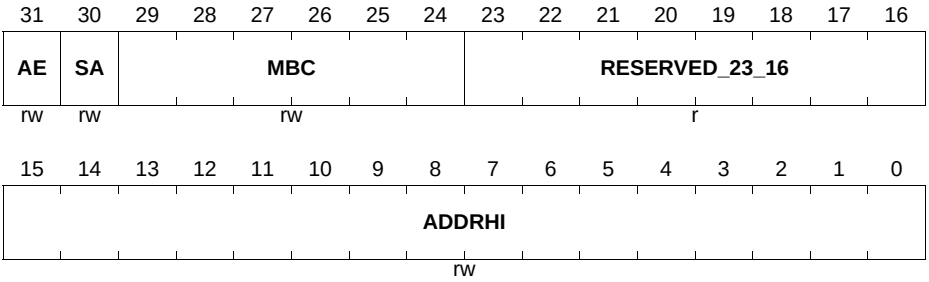
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address29 [31:0]** This field contains the lower 32 bits of the 30th 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address30_High

The MAC Address30 High register holds the upper 16 bits of the 31st 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address30 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address30 Low Register must be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS30_HIGH

Register 540 - MAC Address30 High Register (1870_H) **Reset Value: $0000\ FFFF_H$**



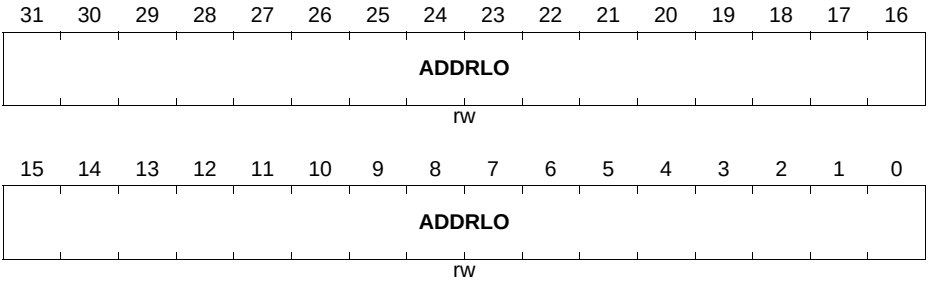
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address30 [47:32]** This field contains the upper 16 bits (47:32) of the 31st 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address30 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address30[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address30[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the 31st MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address30_Low

The MAC Address30 Low register holds the lower 32 bits of the 31st 6-byte MAC address of the station.

ETH_MAC_ADDRESS30_LOW
Register 541 - MAC Address30 Low Register (1874_H) **Reset Value: $FFFF\ FFFF_H$**



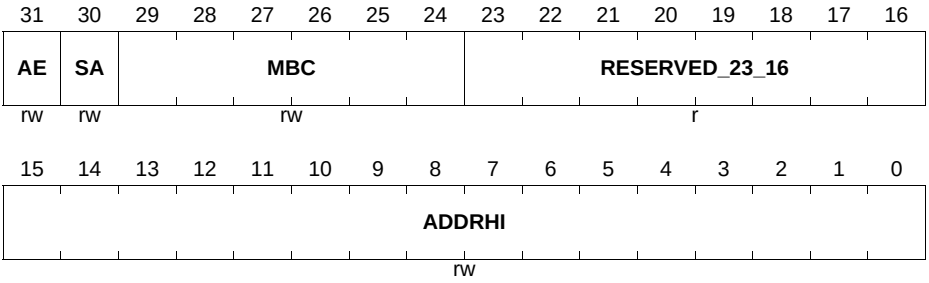
Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address30 [31:0]** This field contains the lower 32 bits of the 31st 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - MAC_Address31_High

The MAC Address31 High register holds the upper 16 bits of the 32nd 6-byte MAC address of the station. If the MAC address registers are configured to be double-synchronized to the (G)MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address31 Low Register are written. For proper synchronization updates, consecutive writes to this MAC Address31 Low Register must be performed after at least four clock cycles in the destination clock domain.

ETH_MAC_ADDRESS31_HIGH

Register 542 - MAC Address31 High Register (1878_H) **Reset Value: $0000\ FFFF_H$**



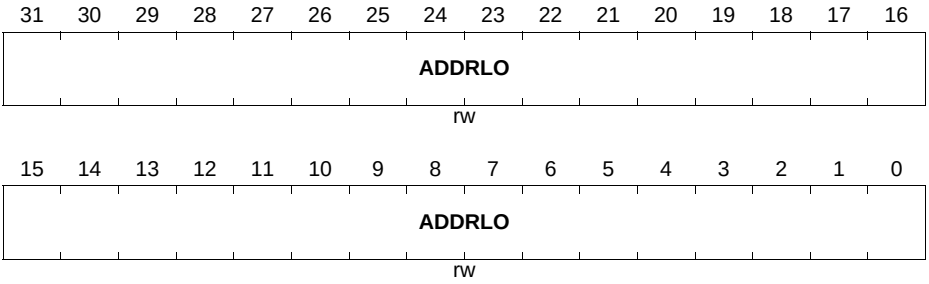
Field	Bits	Type	Description
ADDRHI	[15:0]	rw	**MAC Address31 [47:32]** This field contains the upper 16 bits (47:32) of the 32nd 6-byte MAC address.
RESERVED_23_16	[23:16]	r	**RESERVED_23_16**
MBC	[29:24]	rw	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address31 registers. Each bit controls the masking of the bytes as follows: * Bit 29: Register 18[15:8] * Bit 28: Register 18[7:0] * Bit 27: Register 19[31:24] * ... * Bit 24: Register 19[7:0]

Field	Bits	Type	Description
SA	30	rw	**Source Address** When this bit is set, the MAC Address31[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address31[47:0] is used to compare with the DA fields of the received frame.
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the 32nd MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

32-bit Register - MAC_Address31_Low

The MAC Address31 Low register holds the lower 32 bits of the 32nd 6-byte MAC address of the station.

ETH_MAC_ADDRESS31_LOW
Register 543 - MAC Address31 Low Register ($187C_H$) Reset Value: $FFFF\ FFFF_H$



Field	Bits	Type	Description
ADDRLO	[31:0]	rw	**MAC Address31 [31:0]** This field contains the lower 32 bits of the 32nd 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

32-bit Register - Bus_Mode

The Bus Mode register establishes the bus operating modes for the DMA.

ETH_BUS_MODE

Register 0 - Bus Mode Register (2000_H) Reset Value: $0002\ 0101_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED_31_30		PRWG		TXPR	MB	AAL	PBLx8	USP	RPBL						FB
r		r		rw	rw	rw	rw	rw	rw						rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR		PBL						ATDS	DSL					DA	SWR
rw		rw						rw	rw					rw	rw

Field	Bits	Type	Description
SWR	0	rw	**Software Reset** When this bit is set, the MAC DMA Controller resets the logic and all internal registers of the MAC. It is cleared automatically after the reset operation has completed in all of the DWC_gmac clock domains. Before reprogramming any register of the DWC_gmac, you should read a zero (0) value in this bit . Note: * The Software reset function is driven only by this bit. Bit 0 of Register 64 (Channel 1 Bus Mode Register) or Register 128 (Channel 2 Bus Mode Register) has no impact on the Software reset function. * The reset operation is completed only when all resets in all active clock domains are de-asserted. Therefore, it is essential that all the PHY inputs clocks (applicable for the selected PHY interface) are present for the software reset completion.

Field	Bits	Type	Description
DA	1	rw	**DMA Arbitration Scheme** This bit specifies the arbitration scheme between the transmit and receive paths of Channel 0. * 0: Weighted round-robin with Rx:Tx or Tx:Rx. The priority between the paths is according to the priority specified in bits 15:14 (PR) and priority weights specified in Bit 27 (TXPR). * 1: Fixed priority. The transmit path has priority over receive path when Bit 27 (TXPR) is set. Otherwise, receive path has priority over the transmit path. In the GMAC-AXI configuration, these bits are reserved and read-only (RO).
DSL	[6:2]	rw	**Descriptor Skip Length** This bit specifies the number of Word, Dword, or Lword (depending on the 32-bit, 64-bit, or 128-bit bus) to skip between two unchained descriptors. The address skipping starts from the end of current descriptor to the start of next descriptor. When the DSL value is equal to zero, then the descriptor table is taken as contiguous by the DMA in Ring mode.

Field	Bits	Type	Description
ATDS	7	rw	**Alternate Descriptor Size** When set, the size of the alternate descriptor increases to 32 bytes (8 DWORDS). This is required when the Advanced Timestamp feature or the IPC Full Offload Engine (Type 2) is enabled in the receiver. The enhanced descriptor is not required if the Advanced Timestamp and IPC Full Checksum Offload (Type 2) features are not enabled. In such cases, you can use the 16 bytes descriptor to save 4 bytes of memory. This bit is present only when you select the Alternate Descriptor feature and any one of the following features during core configuration: * Advanced Timestamp feature * IPC Full Checksum Offload Engine (Type 2) feature Otherwise, this bit is reserved and read-only. When reset, the descriptor size reverts back to 4 DWORDs (16 bytes). This bit preserves the backward compatibility for the descriptor size. In versions prior to 3.50a, the descriptor size is 16 bytes for both normal and enhanced descriptor. In version 3.50a, descriptor size is increased to 32 bytes because of the Advanced Timestamp and IPC Full Checksum Offload Engine (Type 2) features.

Field	Bits	Type	Description
PBL	[13:8]	rw	**Programmable Burst Length** These bits indicate the maximum number of beats to be transferred in one DMA transaction. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a Burst transfer on the host bus. PBL can be programmed with permissible values of 1, 2, 4, 8, 16, and 32. Any other value results in undefined behavior. When USP is set high, this PBL value is applicable only for Tx DMA transactions. If the number of beats to be transferred is more than 32, then perform the following steps: 1. Set the PBLx8 mode. 2. Set the PBL. For example, if the maximum number of beats to be transferred is 64, then first set PBLx8 to 1 and then set PBL to 8. The PBL values have the following limitation: The maximum number of possible beats (PBL) is limited by the size of the Tx FIFO and Rx FIFO in the MTL layer and the data bus width on the DMA. The FIFO has a constraint that the maximum beat supported is half the depth of the FIFO, except when specified. For different data bus widths and FIFO sizes, the valid PBL range (including x8 mode) is provided in the following list. If the PBL is common for both transmit and receive DMA, the minimum Rx FIFO and Tx FIFO depths must be considered. Note: In the half-duplex mode, the valid PBL range specified in the following list is applicable only for Tx FIFO. * 32-Bit Data Bus Width - 2 KB and Higher FIFO Depth: All PBL values are supported in the full-duplex mode and half-duplex modes.

Field	Bits	Type	Description
PR	[15:14]	rw	**Priority Ratio** These bits control the priority ratio in the weighted round-robin arbitration between the Rx DMA and Tx DMA. These bits are valid only when Bit 1 (DA) is reset. The priority ratio is Rx:Tx or Tx:Rx depending on whether Bit 27 (TXPR) is reset or set. * 00: The Priority Ratio is 1:1. * 01: The Priority Ratio is 2:1. * 10: The Priority Ratio is 3:1. * 11: The Priority Ratio is 4:1. In the GMAC-AXI configuration, these bits are reserved and read-only (RO).
FB	16	rw	**Fixed Burst** This bit controls whether the AHB or AXI Master interface performs fixed burst transfers or not. When set, the AHB interface uses only SINGLE, INCR4, INCR8, or INCR16 during start of the normal burst transfers. When reset, the AHB or AXI interface uses SINGLE and INCR burst transfer operations. For more information, see Bit 0 (UNDEF) of the AXI Bus Mode register in the GMAC-AXI configuration.
RPBL	[22:17]	rw	**Rx DMA PBL** This field indicates the maximum number of beats to be transferred in one Rx DMA transaction. This is the maximum value that is used in a single block Read or Write. The Rx DMA always attempts to burst as specified in the RPBL bit each time it starts a Burst transfer on the host bus. You can program RPBL with values of 1, 2, 4, 8, 16, and 32. Any other value results in undefined behavior. This field is valid and applicable only when USP is set high.
USP	23	rw	**Use Seperate PBL** When set high, this bit configures the Rx DMA to use the value configured in Bits[22:17] as PBL. The PBL value in Bits[13:8] is applicable only to the Tx DMA operations. When reset to low, the PBL value in Bits[13:8] is applicable for both DMA engines.

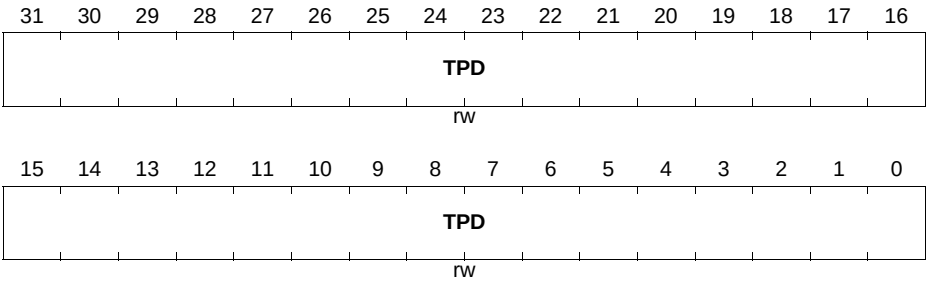
Field	Bits	Type	Description
PBLx8	24	rw	**PBLx8 Mode** When set high, this bit multiplies the programmed PBL value (Bits[22:17] and Bits[13:8]) eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value. Note: This bit function is not backward compatible. Before release 3.50a, this bit was 4xPBL.
AAL	25	rw	**Address Aligned Beats** When this bit is set high and the FB bit is equal to 1, the AHB or AXI interface generates all bursts aligned to the start address LS bits. If the FB bit is equal to 0, the first burst (accessing the data buffer's start address) is not aligned, but subsequent bursts are aligned to the address. This bit is valid only in the GMAC-AHB and GMAC-AXI configuration and is reserved (RO with default value 0) in all other configurations.
MB	26	rw	**Mixed Burst** When this bit is set high and the FB bit is low, the AHB Master interface starts all bursts of length more than 16 with INCR (undefined burst) whereas it reverts to fixed burst transfers (INCRx and SINGLE) for burst length of 16 and less. This bit is valid only in the GMAC-AHB configuration and reserved in all other configuration.
TXPR	27	rw	**Transmit Priority** When set, this bit indicates that the transmit DMA has higher priority than the receive DMA during arbitration for the system-side bus. In the GMAC-AXI configuration, this bit is reserved and read-only (RO).

Field	Bits	Type	Description
PRWG	[29:28]	r	**Channel Priority Weights** This field sets the priority weights for Channel 0 during the round-robin arbitration between the DMA channels for the system bus. * 00: The priority weight is 1. * 01: The priority weight is 2. * 10: The priority weight is 3. * 11: The priority weight is 4. This field is present in all DWC_gmac configurations except GMAC-AXI when you select the AV feature. Otherwise, this field is reserved and read-only (RO).
RESERVED_31_30	[31:30]	r	**RESERVED_31_30**

32-bit Register - Transmit_Poll_Demand

The Transmit Poll Demand register enables the Tx DMA to check whether or not the DMA owns the current descriptor. The Transmit Poll Demand command is given to wake up the Tx DMA if it is in the Suspend mode. The Tx DMA can go into the Suspend mode because of an Underflow error in a transmitted frame or the unavailability of descriptors owned by it. You can give this command anytime and the Tx DMA resets this command when it again starts fetching the current descriptor from host memory.

ETH_TRANSMIT_POLL_DEMAND
Register 1 - Transmit Poll Demand Register (2004_H) **Reset Value: $0000\ 0000_H$**



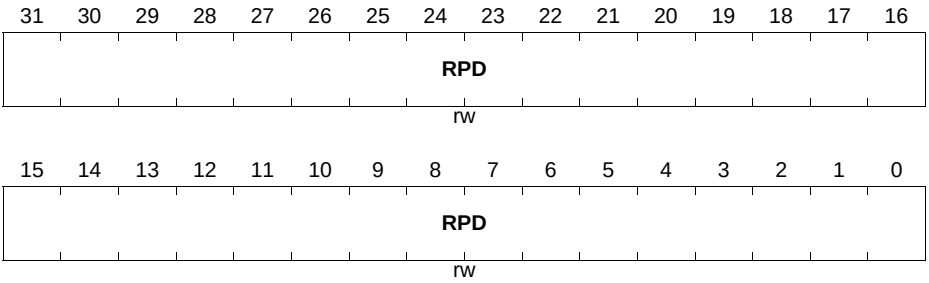
Field	Bits	Type	Description
TPD	[31:0]	rw	**Transmit Poll Demand** When these bits are written with any value, the DMA reads the current descriptor pointed to by Register 18 (Current Host Transmit Descriptor Register). If that descriptor is not available (owned by the Host), the transmission returns to the Suspend state and the Bit 2 (TU) of Register 5 (Status Register) is asserted. If the descriptor is available, the transmission resumes.

32-bit Register - Receive_Poll_Demand

The Receive Poll Demand register enables the receive DMA to check for new descriptors. This command is used to wake up the Rx DMA from the SUSPEND state. The RxDMA can go into the SUSPEND state only because of the unavailability of descriptors it owns.

ETH_RECEIVE_POLL_DEMAND

Register 2 - Receive Poll Demand Register (2008_H) **Reset Value: $0000\ 0000_H$**

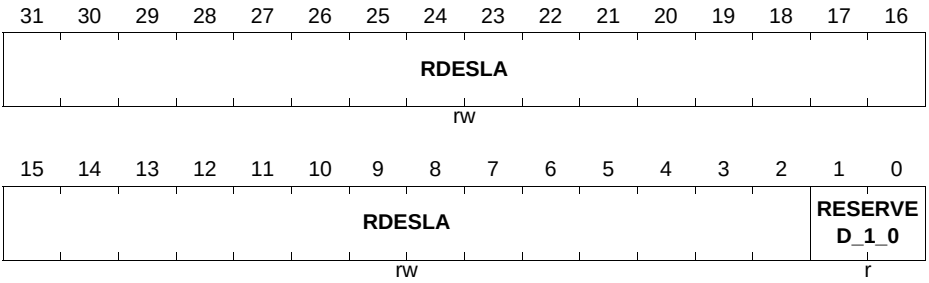


Field	Bits	Type	Description
RPD	[31:0]	rw	**Receive Poll Demand** When these bits are written with any value, the DMA reads the current descriptor pointed to by Register 19 (Current Host Receive Descriptor Register). If that descriptor is not available (owned by the Host), the reception returns to the Suspended state and the Bit 7 (RU) of Register 5 (Status Register) is not asserted. If the descriptor is available, the Rx DMA returns to the active state.

32-bit Register - Receive_Descriptor_List_Address

The Receive Descriptor List Address register points to the start of the Receive Descriptor List. The descriptor lists reside in the host's physical memory space and must be Word, Dword, or Lword-aligned (for 32-bit, 64-bit, or 128-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LS bits low. Writing to this register is permitted only when reception is stopped. When stopped, this register must be written to before the receive Start command is given. You can write to this register only when Rx DMA has stopped, that is, Bit 1 (SR) is set to zero in Register 6 (Operation Mode Register). When stopped, this register can be written with a new descriptor list address. When you set the SR bit to 1, the DMA takes the newly programmed descriptor base address. If this register is not changed when the SR bit is set to 0, then the DMA takes the descriptor address where it was stopped earlier.

ETH_RECEIVE_DESCRIPTOR_LIST_ADDRESS
Register 3 - Receive Descriptor List Address Register ($200C_H$) Reset Value: $0000\ 0000_H$

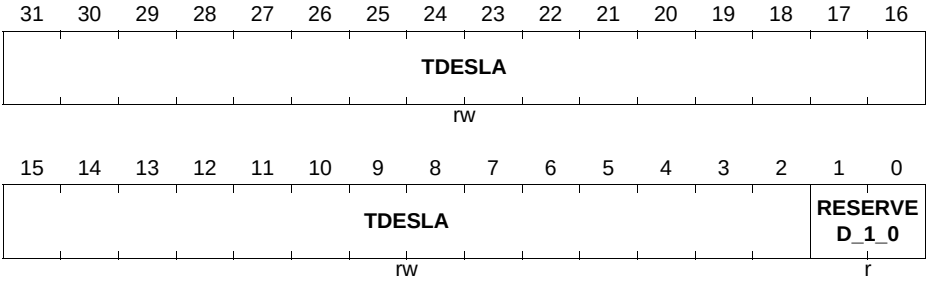


Field	Bits	Type	Description
RESERVED_1_0	[1:0]	r	**RESERVED_1_0**
RDESLA	[31:2]	rw	**Start of Receive List** This field contains the base address of the first descriptor in the Receive Descriptor list. The LSB bits (1:0) for 32-bit bus width are ignored and internally taken as all-zero by the DMA. Therefore, these LSB bits are read-only (RO).

32-bit Register - Transmit_Descriptor_List_Address

The Transmit Descriptor List Address register points to the start of the Transmit Descriptor List. The descriptor lists reside in the host's physical memory space and must be Word, Dword, or Lword-aligned (for 32-bit, 64-bit, or 128-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LSB to low. You can write to this register only when the Tx DMA has stopped, that is, Bit 13 (ST) is set to zero in Register 6 (Operation Mode Register). When stopped, this register can be written with a new descriptor list address. When you set the ST bit to 1, the DMA takes the newly programmed descriptor base address. If this register is not changed when the ST bit is set to 0, then the DMA takes the descriptor address where it was stopped earlier.

ETH_TRANSMIT_DESCRIPTOR_LIST_ADDRESS
Register 4 - Transmit Descriptor List Address Register (2010_H) Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
RESERVED_1_0	[1:0]	r	**RESERVED_1_0**
TDESLA	[31:2]	rw	**Start of Transmit List** This field contains the base address of the first descriptor in the Transmit Descriptor list. The LSB bits (1:0) for 32-bit bus width are ignored and are internally taken as all-zero by the DMA. Therefore, these LSB bits are read-only (RO).

32-bit Register - Status

The Status register contains all status bits that the DMA reports to the host. The Software driver reads this register during an interrupt service routine or polling. Most of the fields in this register cause the host to be interrupted. The bits of this register are not cleared when read. Writing 1'b1 to (unreserved) Bits[16:0] of this register clears these bits and writing 1'b0 has no effect. Each field (Bits[16:0]) can be masked by masking the appropriate bit in Register 7 (Interrupt Enable Register).

ETH_STATUS
Register 5 - Status Register **(2014_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED_31	GLPII	TTI	GPI	GMI	GLI	EB			TS			RS			NIS
r	r	r	r	r	r		r			r			r		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIS	ERI	FBI	RESERVED_12_11		ETI	RWT	RPS	RU	RI	UNF	OVF	TJT	TU	TPS	TI
rw	rw	rw	r		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
TI	0	rw	**Transmit Interrupt** This bit indicates that the frame transmission is complete. When transmission is complete, the Bit 31 (Interrupt on Completion) of TDES1 is reset in the first descriptor, and the specific frame status information is updated in the descriptor.
TPS	1	rw	**Transmit Process Stopped** This bit is set when the transmission is stopped.
TU	2	rw	**Transmit Buffer Unavailable** This bit indicates that the host owns the Next Descriptor in the Transmit List and the DMA cannot acquire it. Transmission is suspended. Bits[22:20] explain the Transmit Process state transitions. To resume processing Transmit descriptors, the host should change the ownership of the descriptor by setting TDES0[31] and then issue a Transmit Poll Demand command.

Field	Bits	Type	Description
TJT	3	rw	**Transmit Jabber Timeout** This bit indicates that the Transmit Jabber Timer expired, which happens when the frame size exceeds 2,048 (10,240 bytes when the Jumbo frame is enabled). When the Jabber Timeout occurs, the transmission process is aborted and placed in the Stopped state. This causes the Transmit Jabber Timeout TDES0[14] flag to assert.
OVF	4	rw	**Receive Overflow** This bit indicates that the Receive Buffer had an Overflow during frame reception. If the partial frame is transferred to the application, the overflow status is set in RDES0[11].
UNF	5	rw	**Transmit Underflow** This bit indicates that the Transmit Buffer had an Underflow during frame transmission. Transmission is suspended and an Underflow Error TDES0[1] is set.
RI	6	rw	**Receive Interrupt** This bit indicates that the frame reception is complete. When reception is complete, the Bit 31 of RDES1 (Disable Interrupt on Completion) is reset in the last Descriptor, and the specific frame status information is updated in the descriptor. The reception remains in the Running state.
RU	7	rw	**Receive Buffer Unavailable** This bit indicates that the host owns the Next Descriptor in the Receive List and the DMA cannot acquire it. The Receive Process is suspended. To resume processing Receive descriptors, the host should change the ownership of the descriptor and issue a Receive Poll Demand command. If no Receive Poll Demand is issued, the Receive Process resumes when the next recognized incoming frame is received. This bit is set only when the previous Receive Descriptor is owned by the DMA.
RPS	8	rw	**Receive Process Stopped** This bit is asserted when the Receive Process enters the Stopped state.

Field	Bits	Type	Description
RWT	9	rw	**Receive Watchdog Timeout** This bit is asserted when a frame with length greater than 2,048 bytes is received (10, 240 when Jumbo Frame mode is enabled).
ETI	10	rw	**Early Transmit Interrupt** This bit indicates that the frame to be transmitted is fully transferred to the MTL Transmit FIFO.
RESERVED_12_11	[12:11]	r	**RESERVED_12_11**
FBI	13	rw	**Fatal Bus Error Interrupt** This bit indicates that a bus error occurred, as described in Bits[25:23]. When this bit is set, the corresponding DMA engine disables all of its bus accesses.
ERI	14	rw	**Early Receive Interrupt** This bit indicates that the DMA had filled the first data buffer of the packet. Bit 6 (RI) of this register automatically clears this bit.
AIS	15	rw	**Abnormal Interrupt Summary** Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in Register 7 (Interrupt Enable Register): * Register 5[1]: Transmit Process Stopped * Register 5[3]: Transmit Jabber Timeout * Register 5[4]: Receive FIFO Overflow * Register 5[5]: Transmit Underflow * Register 5[7]: Receive Buffer Unavailable * Register 5[8]: Receive Process Stopped * Register 5[9]: Receive Watchdog Timeout * Register 5[10]: Early Transmit Interrupt * Register 5[13]: Fatal Bus Error Only unmasked bits affect the Abnormal Interrupt Summary bit. This is a sticky bit and must be cleared each time a corresponding bit, which causes AIS to be set, is cleared.

Field	Bits	Type	Description
NIS	16	rw	**Normal Interrupt Summary** Normal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in Register 7 (Interrupt Enable Register): * Register 5[0]: Transmit Interrupt * Register 5[2]: Transmit Buffer Unavailable * Register 5[6]: Receive Interrupt * Register 5[14]: Early Receive Interrupt Only unmasked bits (interrupts for which interrupt enable is set in Register 7) affect the Normal Interrupt Summary bit. This is a sticky bit and must be cleared (by writing 1 to this bit) each time a corresponding bit, which causes NIS to be set, is cleared.
RS	[19:17]	r	**Received Process State** This field indicates the Receive DMA FSM state. This field does not generate an interrupt. * 3'b000: Stopped: Reset or Stop Receive Command issued * 3'b001: Running: Fetching Receive Transfer Descriptor * 3'b010: Reserved for future use * 3'b011: Running: Waiting for receive packet * 3'b100: Suspended: Receive Descriptor Unavailable * 3'b101: Running: Closing Receive Descriptor * 3'b110: TIME_STAMP write state * 3'b111: Running: Transferring the receive packet data from receive buffer to host memory

Field	Bits	Type	Description
TS	[22:20]	r	**Transmit Process State** This field indicates the Transmit DMA FSM state. This field does not generate an interrupt. * 3'b000: Stopped; Reset or Stop Transmit Command issued * 3'b001: Running; Fetching Transmit Transfer Descriptor * 3'b010: Running; Waiting for status * 3'b011: Running; Reading Data from host memory buffer and queuing it to transmit buffer (Tx FIFO) * 3'b100: TIME_STAMP write state * 3'b101: Reserved for future use * 3'b110: Suspended; Transmit Descriptor Unavailable or Transmit Buffer Underflow * 3'b111: Running; Closing Transmit Descriptor
EB	[25:23]	r	**Error Bits** This field indicates the type of error that caused a Bus Error, for example, error response on the AHB or AXI interface. This field is valid only when Bit 13 (FBI) is set. This field does not generate an interrupt. * Bit 23 - 1'b1: Error during data transfer by the Tx DMA - 1'b0: Error during data transfer by the Rx DMA * Bit 24 - 1'b1: Error during read transfer - 1'b0: Error during write transfer * Bit 25 - 1'b1: Error during descriptor access - 1'b0: Error during data buffer access
GLI	26	r	**GMAC Line interface Interrupt** This bit reflects an interrupt event in the PCS (link change and AN complete), SMII (link change), or RGMII (link change) interface block of the DWC_gmac. The software must read the corresponding registers (Register 49 for PCS or Register 54 for SMII or RGMII) in the DWC_gmac to get the exact cause of the interrupt and clear the source of interrupt to make this bit as 1'b0. The interrupt signal from the DWC_gmac subsystem (sbd_intr_o) is high when this bit is high.

Field	Bits	Type	Description
GMI	27	r	**GMAC MMC Interrupt** This bit reflects an interrupt event in the MMC module of the DWC_gmac. The software must read the corresponding registers in the DWC_gmac to get the exact cause of interrupt and clear the source of interrupt to make this bit as 1'b0. The interrupt signal from the DWC_gmac subsystem (sbd_intr_o) is high when this bit is high. This bit is applicable only when the MAC Management Counters (MMC) are enabled. Otherwise, this bit is reserved.
GPI	28	r	**GMAC PMT Interrupt** This bit indicates an interrupt event in the PMT module of the DWC_gmac. The software must read the PMT Control and Status Register in the MAC to get the exact cause of interrupt and clear its source to reset this bit to 1'b0. The interrupt signal from the DWC_gmac subsystem (sbd_intr_o) is high when this bit is high. This bit is applicable only when the Power Management feature is enabled. Otherwise, this bit is reserved. Note: This interrupt is different from the pmt_intr_o interrupt.
TTI	29	r	**Timestamp Trigger Interrupt** This bit indicates an interrupt event in the Timestamp Generator block of DWC_gmac. The software must read the corresponding registers in the DWC_gmac to get the exact cause of interrupt and clear its source to reset this bit to 1'b0. When this bit is high, the interrupt signal from the DWC_gmac subsystem (sbd_intr_o) is high. This bit is applicable only when the IEEE 1588 Timestamp feature is enabled. Otherwise, this bit is reserved.

Field	Bits	Type	Description
GLPII	30	r	**GMAC LPI Interrupt (for Channel 0)** This bit indicates an interrupt event in the LPI logic of the DWC_gmac. To reset this bit to 1'b0, the software must read the corresponding registers in the DWC_gmac to get the exact cause of the interrupt and clear its source. Note: GLPII status is given only in Channel 0 DMA register and is applicable only when the Energy Efficient Ethernet feature is enabled. Otherwise, this bit is reserved.When this bit is high, the interrupt signal from the MAC (sbd_intr_o) is high.
RESERVED_31	31	r	**RESERVED_31**

32-bit Register - Operation_Mode

The Operation Mode register establishes the Transmit and Receive operating modes and commands. This register should be the last CSR to be written as part of the DMA initialization. This register is also present in the GMAC-MTL configuration with unused and reserved bits 24, 13, 2, and 1.

ETH_OPERATION_MODE

Register 6 - Operation Mode Register (2018_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED_31_27					DT	RSF	DFF	RFA_2	RFD_2	TSF	FTF	RESERVED_19_17			TTC
r					rw	rw	rw	r	r	rw	rw	r			rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTC		ST	RFD		RFA		EFC	FEF	FUF	RESERVED_5	RTC		OSF	SR	RESERVED_0
rw		rw	rw		rw		rw	rw	rw	r	rw		rw	rw	r

Field	Bits	Type	Description
RESERVED_0	0	r	**RESERVED_0**

Field	Bits	Type	Description
SR	1	rw	**Start or Stop Receive** When this bit is set, the Receive process is placed in the Running state. The DMA attempts to acquire the descriptor from the Receive list and processes the incoming frames. The descriptor acquisition is attempted from the current position in the list, which is the address set by Register 3 (Receive Descriptor List Address Register) or the position retained when the Receive process was previously stopped. If the DMA does not own the descriptor, reception is suspended and Bit 7 (Receive Buffer Unavailable) of Register 5 (Status Register) is set. The Start Receive command is effective only when the reception has stopped. If the command is issued before setting Register 3 (Receive Descriptor List Address Register), the DMA behavior is unpredictable. When this bit is cleared, the Rx DMA operation is stopped after the transfer of the current frame. The next descriptor position in the Receive list is saved and becomes the current position after the Receive process is restarted. The Stop Receive command is effective only when the Receive process is in either the Running (waiting for receive packet) or in the Suspended state.
OSF	2	rw	**Operate on Second Frame** When this bit is set, it instructs the DMA to process the second frame of the Transmit data even before the status for the first frame is obtained.
RTC	[4:3]	rw	**Receive Threshold Control** These two bits control the threshold level of the MTL Receive FIFO. Transfer (request) to DMA starts when the frame size within the MTL Receive FIFO is larger than the threshold. In addition, full frames with length less than the threshold are transferred automatically. The value of 11 is not applicable if the configured Receive FIFO size is 128 bytes. These bits are valid only when the RSF bit is zero, and are ignored when the RSF bit is set to 1. * 00: 64 * 01: 32 * 10 : 96 * 11: 128

Field	Bits	Type	Description
RESERVED_5	5	r	**RESERVED_5**
FUF	6	rw	**Forward Undersized Good Frames** When set, the Rx FIFO forwards Undersized frames (frames with no Error and length less than 64 bytes) including pad-bytes and CRC. When reset, the Rx FIFO drops all frames of less than 64 bytes, unless a frame is already transferred because of the lower value of Receive Threshold, for example, RTC = 01.
FEF	7	rw	**Forward Error Frames** When this bit is reset, the Rx FIFO drops frames with error status (CRC error, collision error, GMII_ER, giant frame, watchdog timeout, or overflow). However, if the start byte (write) pointer of a frame is already transferred to the read controller side (in Threshold mode), then the frame is not dropped. In the GMAC-MTL configuration in which the Frame Length FIFO is also enabled during core configuration, the Rx FIFO drops the error frames if that frame's start byte is not transferred (output) on the ARI bus. When the FEF bit is set, all frames except runt error frames are forwarded to the DMA. If the Bit 25 (RSF) is set and the Rx FIFO overflows when a partial frame is written, then the frame is dropped irrespective of the FEF bit setting. However, if the Bit 25 (RSF) is reset and the Rx FIFO overflows when a partial frame is written, then a partial frame may be forwarded to the DMA.
EFC	8	rw	**Enable HW Flow Control** When this bit is set, the flow control signal operation based on the fill-level of Rx FIFO is enabled. When reset, the flow control operation is disabled. This bit is not used (reserved and always reset) when the Rx FIFO is less than 4 KB.

Field	Bits	Type	Description
RFA	[10:9]	rw	**Threshold for Activating Flow Control (in half-duplex and full-duplex)** These bits control the threshold (Fill level of Rx FIFO) at which the flow control is activated. - 00: Full minus 1 KB, that is, FULL - 1KB - 01: Full minus 2 KB, that is, FULL - 2KB - 10: Full minus 3 KB, that is, FULL - 3KB - 11: Full minus 4 KB, that is, FULL - 4KB These values only apply to Rx FIFOs of 4 KB or more when the EFC bit is set high. If the Rx FIFO is 8 KB or more, an additional bit (RFA_2) is used for more threshold levels as described in Bit 23. These bits are reserved and read-only when the depth of Rx FIFO is less than 4 KB.
RFD	[12:11]	rw	**Threshold for Deactivating Flow Control (in half-duplex and full-duplex)** These bits control the threshold (Fill-level of Rx FIFO) at which the flow control is de-asserted after activation. - 00: Full minus 1 KB, that is, FULL - 1KB - 01: Full minus 2 KB, that is, FULL - 2KB - 10: Full minus 3 KB, that is, FULL - 3KB - 11: Full minus 4 KB, that is, FULL - 4KB The de-assertion is effective only after flow control is asserted. If the Rx FIFO is 8 KB or more, an additional bit (RFD_2) is used for more threshold levels as described in Bit 22. These bits are reserved and read-only when the Rx FIFO depth is less than 4 KB.

Field	Bits	Type	Description
ST	13	rw	**Start or Stop Transmission Command** When this bit is set, transmission is placed in the Running state, and the DMA checks the Transmit List at the current position for a frame to be transmitted. Descriptor acquisition is attempted either from the current position in the list, which is the Transmit List Base Address set by Register 4 (Transmit Descriptor List Address Register), or from the position retained when transmission was stopped previously. If the DMA does not own the current descriptor, transmission enters the Suspended state and Bit 2 (Transmit Buffer Unavailable) of Register 5 (Status Register) is set. The Start Transmission command is effective only when transmission is stopped. If the command is issued before setting Register 4 (Transmit Descriptor List Address Register), then the DMA behavior is unpredictable. When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current frame. The Next Descriptor position in the Transmit List is saved, and it becomes the current position when transmission is restarted. To change the list address, you need to program Register 4 (Transmit Descriptor List Address Register) with a new value when this bit is reset. The new value is considered when this bit is set again. The stop transmission command is effective only when the transmission of the current frame is complete or the transmission is in the Suspended state.

Field	Bits	Type	Description
TTC	[16:14]	rw	**Transmit Threshold Control** These bits control the threshold level of the MTL Transmit FIFO. Transmission starts when the frame size within the MTL Transmit FIFO is larger than the threshold. In addition, full frames with a length less than the threshold are also transmitted. These bits are used only when Bit 21 (TSF) is reset. * 000: 64 * 001: 128 * 010: 192 * 011: 256 * 100: 40 * 101: 32 * 110: 24 * 111: 16
RESERVED_19_17	[19:17]	r	**RESERVED_19_17**
FTF	20	rw	**Flush Transmit FIFO** When this bit is set, the transmit FIFO controller logic is reset to its default values and thus all data in the Tx FIFO is lost or flushed. This bit is cleared internally when the flushing operation is completed. The Operation Mode register should not be written to until this bit is cleared. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt frame transmission. Note: The flush operation is complete only when the Tx FIFO is emptied of its contents and all the pending Transmit Status of the transmitted frames are accepted by the host. To complete this flush operation, the PHY transmit clock (clk_tx_i) is required to be active.
TSF	21	rw	**Transmit Store and Forward** When this bit is set, transmission starts when a full frame resides in the MTL Transmit FIFO. When this bit is set, the TTC values specified in Bits[16:14] are ignored. This bit should be changed only when the transmission is stopped.

Field	Bits	Type	Description
RFD_2	22	r	**MSB of Threshold for Deactivating Flow Control** If the DWC_gmac is configured for Rx FIFO size of 8 KB or more, this bit (when set) provides additional threshold levels for deactivating the flow control in both half-duplex and full-duplex modes. This bit (as Most Significant Bit) along with the RFD (Bits[12:11]) gives the following thresholds for deactivating flow control: * 100: Full minus 5 KB, that is, FULL - 5KB * 101: Full minus 6 KB, that is, FULL - 6KB * 110: Full minus 7 KB, that is, FULL - 7KB * 111: Reserved This bit is reserved (and RO) if the Rx FIFO is 4 KB or less deep.
RFA_2	23	r	**MSB of Threshold for Activating Flow Control** If the DWC_gmac is configured for an Rx FIFO depth of 8 KB or more, this bit (when set) provides additional threshold levels for activating the flow control in both half-duplex and full-duplex modes. This bit (as Most Significant Bit) along with the RFA (Bits[10:9]) gives the following thresholds for activating flow control: * 100: Full minus 5 KB, that is, FULL - 5KB * 101: Full minus 6 KB, that is, FULL - 6KB * 110: Full minus 7 KB, that is, FULL - 7KB * 111: Reserved This bit is reserved (and RO) if the Rx FIFO is 4 KB or less deep.
DFF	24	rw	**Disable Flushing of Received Frames** When this bit is set, the Rx DMA does not flush any frames because of the unavailability of receive descriptors or buffers as it does normally when this bit is reset. This bit is reserved (and RO) in the GMAC-MTL configuration.

Field	Bits	Type	Description
RSF	25	rw	**Receive Store and Forward** When this bit is set, the MTL reads a frame from the Rx FIFO only after the complete frame has been written to it, ignoring the RTC bits. When this bit is reset, the Rx FIFO operates in the cut-through mode, subject to the threshold specified by the RTC bits.
DT	26	rw	**Disable Dropping of TCP/IP Checksum Error Frames** When this bit is set, the MAC does not drop the frames which only have errors detected by the Receive Checksum Offload engine. Such frames do not have any errors (including FCS error) in the Ethernet frame received by the MAC but have errors only in the encapsulated payload. When this bit is reset, all error frames are dropped if the FEF bit is reset. If the IPC Full Checksum Offload Engine (Type 2) is disabled, this bit is reserved (RO with value 1'b0).
RESERVED_31_27	[31:27]	r	**RESERVED_31_27**

32-bit Register - Interrupt_Enable

The Interrupt Enable register enables the interrupts reported by Register 5 (Status Register). Setting a bit to 1'b1 enables a corresponding interrupt. After a hardware or software reset, all interrupts are disabled.

ETH_INTERRUPT_ENABLE
Register 7 - Interrupt Enable Register (201C$_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED_31_17															NIE
r															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIE	ERE	FBE	RESERVED_12_11		ETE	RWE	RSE	RUE	RIE	UNE	OVE	TJE	TUE	TSE	TIE
rw	rw	rw	r		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
TIE	0	rw	**Transmit Interrupt Enable** When this bit is set with Normal Interrupt Summary Enable (Bit 16), the Transmit Interrupt is enabled. When this bit is reset, the Transmit Interrupt is disabled.
TSE	1	rw	**Transmit Stopped Enable** When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Transmission Stopped Interrupt is enabled. When this bit is reset, the Transmission Stopped Interrupt is disabled.
TUE	2	rw	**Transmit Buffer Unvailable Enable** When this bit is set with Normal Interrupt Summary Enable (Bit 16), the Transmit Buffer Unavailable Interrupt is enabled. When this bit is reset, the Transmit Buffer Unavailable Interrupt is disabled.
TJE	3	rw	**Transmit Jabber Timeout Enable** When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Transmit Jabber Timeout Interrupt is enabled. When this bit is reset, the Transmit Jabber Timeout Interrupt is disabled.

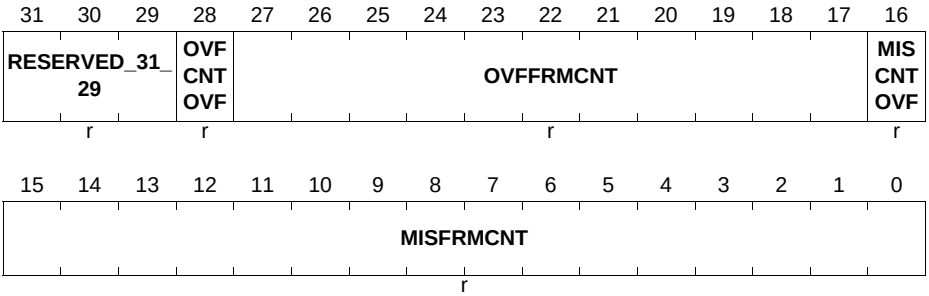
Field	Bits	Type	Description
OVE	4	rw	**Overflow Interrupt Enable** When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Receive Overflow Interrupt is enabled. When this bit is reset, the Overflow Interrupt is disabled.
UNE	5	rw	**Underflow Interrupt Enable** When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Transmit Underflow Interrupt is enabled. When this bit is reset, the Underflow Interrupt is disabled.
RIE	6	rw	**Receive Interrupt Enable** When this bit is set with Normal Interrupt Summary Enable (Bit 16), the Receive Interrupt is enabled. When this bit is reset, the Receive Interrupt is disabled.
RUE	7	rw	**Receive Buffer Unavailable Enable** When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Receive Buffer Unavailable Interrupt is enabled. When this bit is reset, the Receive Buffer Unavailable Interrupt is disabled.
RSE	8	rw	**Receive Stopped Enable** When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Receive Stopped Interrupt is enabled. When this bit is reset, the Receive Stopped Interrupt is disabled.
RWE	9	rw	**Receive Watchdog Timeout Enable** When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Receive Watchdog Timeout Interrupt is enabled. When this bit is reset, the Receive Watchdog Timeout Interrupt is disabled.
ETE	10	rw	**Early Transmit Interrupt Enable** When this bit is set with an Abnormal Interrupt Summary Enable (Bit 15), the Early Transmit Interrupt is enabled. When this bit is reset, the Early Transmit Interrupt is disabled.
RESERVED_12_11	[12:11]	r	**RESERVED_12_11**

Field	Bits	Type	Description
FBE	13	rw	**Fatal Bus Error Enable** When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Fatal Bus Error Interrupt is enabled. When this bit is reset, the Fatal Bus Error Enable Interrupt is disabled.
ERE	14	rw	**Early Receive Interrupt Enable** When this bit is set with Normal Interrupt Summary Enable (Bit 16), the Early Receive Interrupt is enabled. When this bit is reset, the Early Receive Interrupt is disabled.
AIE	15	rw	**Abnormal Interrupt Summary Enable** When this bit is set, abnormal interrupt summary is enabled. When this bit is reset, the abnormal interrupt summary is disabled. This bit enables the following interrupts in Register 5 (Status Register): * Register 5[1]: Transmit Process Stopped * Register 5[3]: Transmit Jabber Timeout * Register 5[4]: Receive Overflow * Register 5[5]: Transmit Underflow * Register 5[7]: Receive Buffer Unavailable * Register 5[8]: Receive Process Stopped * Register 5[9]: Receive Watchdog Timeout * Register 5[10]: Early Transmit Interrupt * Register 5[13]: Fatal Bus Error
NIE	16	rw	**Normal Interrupt Summary Enable** When this bit is set, normal interrupt summary is enabled. When this bit is reset, normal interrupt summary is disabled. This bit enables the following interrupts in Register 5 (Status Register): * Register 5[0]: Transmit Interrupt * Register 5[2]: Transmit Buffer Unavailable * Register 5[6]: Receive Interrupt * Register 5[14]: Early Receive Interrupt
RESERVED_31_17	[31:17]	r	**RESERVED_31_17**

32-bit Register - Missed_Frame_And_Buffer_Overflow_Counter

The DMA maintains two counters to track the number of frames missed during reception. This register reports the current value of the counter. The counter is used for diagnostic purposes. Bits[15:0] indicate missed frames because of the host buffer being unavailable. Bits[27:17] indicate missed frames because of buffer overflow conditions (MTL and MAC) and runt frames (good frames of less than 64 bytes) dropped by the MTL.

ETH_MISSED_FRAME_AND_BUFFER_OVERFLOW_COUNTER
Register 8 - Missed Frame and Buffer Overflow Counter Register (2020_H) Reset Value: 0000 0000_H



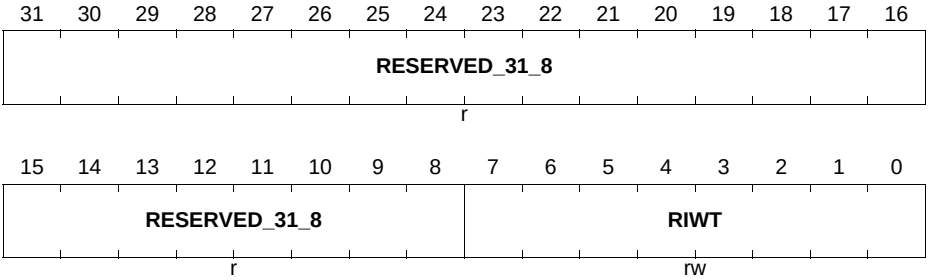
Field	Bits	Type	Description
MISFRMCNT	[15:0]	r	**MISFRMCNT** This field indicates the number of frames missed by the controller because of the Host Receive Buffer being unavailable. This counter is incremented each time the DMA discards an incoming frame. The counter is cleared when this register is read with mci_be_i[0] at 1'b1.
MISCNTOVF	16	r	**MISCNTOVF** Overflow bit for Missed Frame Counter
OVFFRMCNT	[27:17]	r	**OVFFRMCNT** This field indicates the number of frames missed by the application. This counter is incremented each time the MTL asserts the sideband signal mtl_rxoverflow_o. The counter is cleared when this register is read with mci_be_i[2] at 1'b1.

Field	Bits	Type	Description
OVFCNTOVF	28	r	**OVFCNTOVF** Overflow bit for FIFO Overflow Counter
RESERVED_31_29	[31:29]	r	**RESERVED_31_29**

32-bit Register - Receive_Interrupt_Watchdog_Timer

This register, when written with non-zero value, enables the watchdog timer for the Receive Interrupt (Bit 6) of Register 5 (Status Register)

ETH_RECEIVE_INTERRUPT_WATCHDOG_TIMER
Register 9 - Receive Interrupt Watchdog Timer Register (2024_H)Reset Value: 0000 0000_H

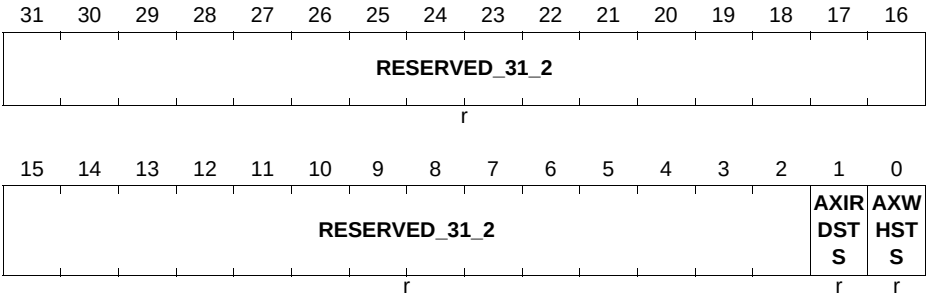


Field	Bits	Type	Description
RIWT	[7:0]	rw	**RI Watchdog Timer Count** This bit indicates the number of system clock cycles multiplied by 256 for which the watchdog timer is set. The watchdog timer gets triggered with the programmed value after the Rx DMA completes the transfer of a frame for which the RI status bit is not set because of the setting in the corresponding descriptor RDES1[31]. When the watchdog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when the RI bit is set high because of automatic setting of RI as per RDES1[31] of any received frame.
RESERVED_31_8	[31:8]	r	**RESERVED_31_8**

32-bit Register - AHB_or_AXI_Status

This register provides the active status of the AHB master interface or AXI interface's read and write channels. This register is present and valid only in the GMAC-AHB and GMAC-AXI configurations. This register is useful for debugging purposes. In addition, this register is valid only in the Channel 0 DMA when multiple channels are present in the AV mode.

ETH_AHB_OR_AXI_STATUS
Register 11 - AHB or AXI Status Register ($202C_H$) **Reset Value: $0000\ 0000_H$**

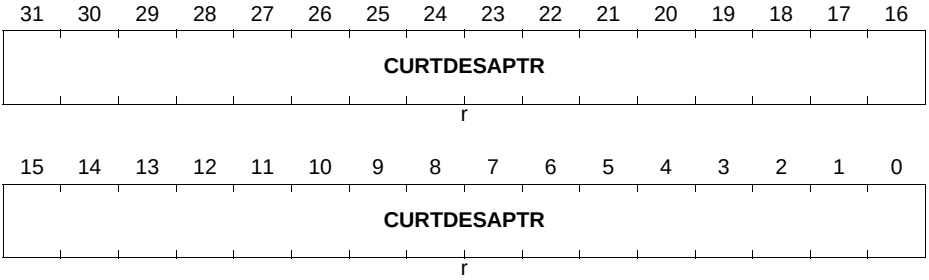


Field	Bits	Type	Description
AXWHSTS	0	r	**AXI Master Write Channel or AHB Master Status** When high, it indicates that AXI Master's write channel is active and transferring data in the GMAC-AXI configuration. In the GMAC-AHB configuration, it indicates that the AHB master interface FSMs are in the non-idle state.
AXIRDSTS	1	r	**AXI Master Read Channel Status** When high, it indicates that AXI Master's read channel is active and transferring data.
RESERVED_31_2	[31:2]	r	**RESERVED_31_2**

32-bit Register - Current_Host_Transmit_Descriptor

The Current Host Transmit Descriptor register points to the start address of the current Transmit Descriptor read by the DMA.

ETH_CURRENT_HOST_TRANSMIT_DESCRIPTOR
Register 18 - Current Host Transmit Descriptor Register (2048_H)Reset Value: 0000 0000_H

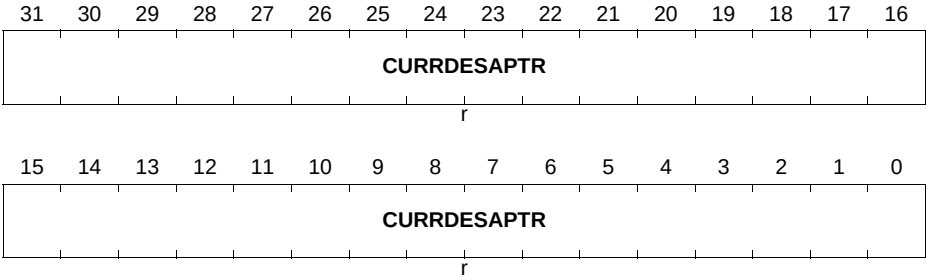


Field	Bits	Type	Description
CURTDESAPTR	[31:0]	r	**Host Transmit Descriptor Address Pointer** Cleared on Reset. Pointer updated by the DMA during operation.

32-bit Register - Current_Host_Receive_Descriptor

The Current Host Receive Descriptor register points to the start address of the current Receive Descriptor read by the DMA.

ETH_CURRENT_HOST_RECEIVE_DESCRIPTOR
Register 19 - Current Host Receive Descriptor Register ($204C_H$) Reset Value: $0000\ 0000_H$

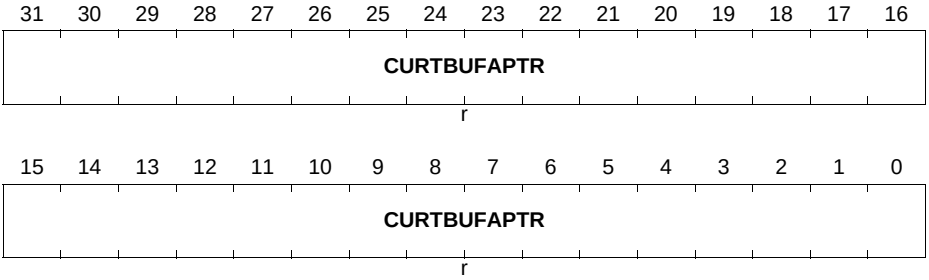


Field	Bits	Type	Description
CURRDESAPTR	[31:0]	r	**Host Receive Descriptor Address Pointer** Cleared on Reset. Pointer updated by the DMA during operation.

32-bit Register - Current_Host_Transmit_Buffer_Address

The Current Host Transmit Buffer Address register points to the current Transmit Buffer Address being read by the DMA.

ETH_CURRENT_HOST_TRANSMIT_BUFFER_ADDRESS
Register 20 - Current Host Transmit Buffer Address Register (2050_H) Reset Value: $0000\ 0000_H$

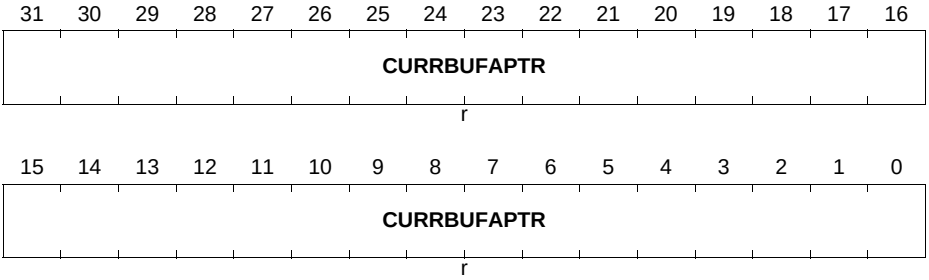


Field	Bits	Type	Description
CURTBUFAPTR	[31:0]	r	**Host Transmit Buffer Address Pointer** Cleared on Reset. Pointer updated by the DMA during operation.

32-bit Register - Current_Host_Receive_Buffer_Address

The Current Host Receive Buffer Address register points to the current Receive Buffer address being read by the DMA.

ETH_CURRENT_HOST_RECEIVE_BUFFER_ADDRESS
Register 21 - Current Host Receive Buffer Address Register (2054_H) Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
CURRBUFAPTR	[31:0]	r	**Host Receive Buffer Address Pointer** Cleared on Reset. Pointer updated by the DMA during operation.

32-bit Register - HW_Feature

This register indicates the presence of the optional features or functions of the DWC_gmac. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks. Note: All bits are set or reset as per the selection of features during the DWC_gmac configuration.

ETH_HW_FEATURE

Register 22 - HW Feature Register (2058$_H$) **Reset Value:030D 2F35$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESERVED_31	ACTPHYIF			SAVLANINS	FLEXIPPSEN	INTTSEN	ENHDESSEL	TXCHCNT		RXCHCNT		RXFIFOSIZE	RXTYP2COE	RXTYP1COE	TXCOESEL
r	r			r	r	r	r	r		r		rw	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AVSEL	EEESEL	TSVER2SEL	TSVER1SEL	MMCSEL	MGKSEL	RWKSEL	SMASEL	L3L4FLTREN	PCSSEL	ADDMACADRSEL	HASHSEL	EXTHASHEN	HDSEL	GMIISEL	MIISEL
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
MIISEL	0	r	**MIISEL** 10 or 100 Mbps support
GMIISEL	1	r	**GMIISEL** 1000 Mbps support
HDSEL	2	r	**HDSEL** Half-Duplex support
EXTHASHEN	3	r	**EXTHASHEN** Expanded DA Hash Filter
HASHSEL	4	r	**HASHSEL** HASH Filter
ADDMACADRSEL	5	r	**ADDMACADRSEL** Multiple MAC Address Registers
PCSSEL	6	r	**PCSSEL** PCS registers (TBI, SGMII, or RTBI PHY interface)

Field	Bits	Type	Description
L3L4FLTREN	7	r	**L3L4FLTREN** Layer 3 and Layer 4 Filter Feature
SMASEL	8	r	**SMASEL** SMA (MDIO) Interface
RWKSEL	9	r	**RWKSEL** PMT Remote Wakeup
MGKSEL	10	r	**MGKSEL** PMT Magic Packet
MMCSEL	11	r	**MMCSEL** RMON Module
TSVER1SEL	12	r	**TSVER1SEL** Only IEEE 1588-2002 Timestamp
TSVER2SEL	13	r	**TSVER2SEL** IEEE 1588-2008 Advanced Timestamp
EEESEL	14	r	**EEESEL** Energy Efficient Ethernet
AVSEL	15	r	**AVSEL** AV Feature
TXCOESEL	16	r	**TXCOESEL** Checksum Offload in Tx
RXTYP1COE	17	r	**RXTYP1COE** IP Checksum Offload (Type 1) in Rx
RXTYP2COE	18	r	**RXTYP2COE** IP Checksum Offload (Type 2) in Rx
RXFIFOSIZE	19	rw	**RXFIFOSIZE** Rx FIFO > 2,048 Bytes
RXCHCNT	[21:20]	r	**RXCHCNT** Number of additional Rx channels
TXCHCNT	[23:22]	r	**TXCHCNT** Number of additional Tx channels
ENHDESSEL	24	r	**ENHDESSEL** Alternate (Enhanced Descriptor)
INTTSEN	25	r	**INTTSEN** Timestamping with Internal System Time

Field	Bits	Type	Description
FLEXIPPSEN	26	r	**FLEXIPPSEN** Flexible Pulse-Per-Second Output
SAVLANINS	27	r	**SAVLANINS** Source Address or VLAN Insertion
ACTPHYIF	[30:28]	r	**Active or Selected PHY interface** When you have multiple PHY interfaces in your configuration, this field indicates the sampled value of phy_intf_sel_i during reset de-assertion * 0000: GMII or MII * 0001: RGMII * 0010: SGMII * 0011: TBI * 0100: RMII * 0101: RTBI * 0110: SMII * 0111: RevMII * All Others: Reserved
RESERVED_31	31	r	**RESERVED_31**

30.4 Descriptors

This chapter comprises the following subsections:

- **Normal Descriptor Formats**
- **Alternate or Enhanced Descriptors**

Note: The Ethernet MAC *in* TC21x/TC22x/TC23x *does not support Normal Descriptor Format as it is configured to support IEEE1588 PTP time stamping! Nevertheless parts of the explanations in the chapter* **Chapter 1.7.1** *are required for complete understanding.*

30.4.1 Normal Descriptor Formats

The DMA in the Ethernet subsystem transfers data based on a linked list of descriptors, as explained in **DMA Controller**. The default descriptor formats (common for both Receive and Transmit Descriptors) are shown in **Figure 30-24**, and field descriptions are provided in **Chapter 1.7.1.1** to **Chapter 1.7.1.2**.

Note:

14. *All descriptions in* ***Chapter 1.7.1.1*** *and* ***Chapter 1.7.1.2*** *correspond to the default descriptor format. The alternate enhanced descriptor format is discussed in* ***Chapter 1.7.2*** *"Alternate or Enhanced Descriptors" on* ***Page 1-680****.*
15. *Changes to the default descriptor format when IEEE1588 time stamping is enabled are described in* ***Chapter 1.7.1.3 "Descriptor Format With IEEE 1588 Time Stamping Enabled" on Page 1-676****".*

Each descriptor contains two buffers, two byte-count buffers, and two address pointers, which enable the adapter port to be compatible with various types of memory management schemes.

The descriptor addresses must be aligned to the bus width used (Word/Dword/Lword for 32-bit buses).

Figure 30-24 shows the descriptor format in a 32-bit data bus.

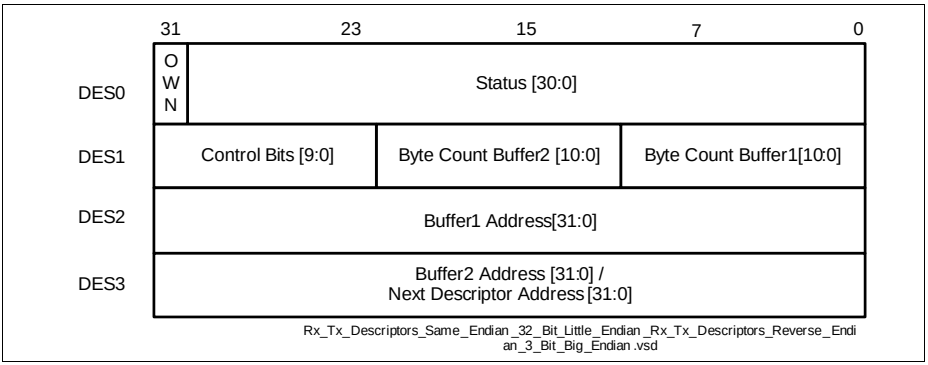


Figure 30-24 Rx/Tx Descriptors for 32-Bit Data Bus

30.4.1.1 Receive Descriptor

The GMAC Subsystem requires at least two descriptors when receiving a frame. The Receive state machine of the DMA (in the GMAC Subsystem) always attempts to acquire an extra descriptor in anticipation of an incoming frame. (The size of the incoming frame is unknown). Before the RxDMA closes a descriptor, it will attempt to acquire the next descriptor even if no frames are received.

In a single descriptor (receive) system, the subsystem will generate a descriptor error if the receive buffer is unable to accommodate the incoming frame and the next descriptor is not owned by the DMA. Thus, the Host is forced to increase either its descriptor pool or the buffer size. Otherwise, the subsystem starts dropping all incoming frames.

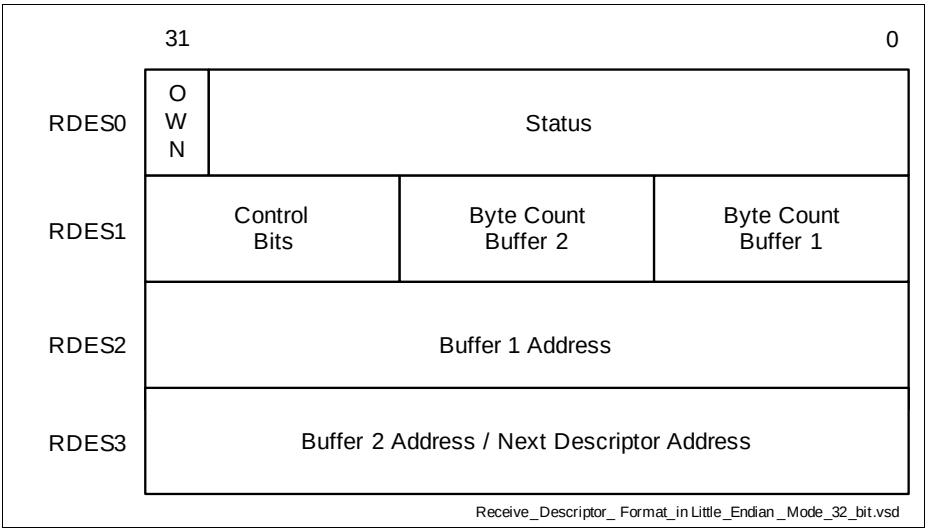


Figure 30-25 Receive Descriptor Format in Little-Endian Mode With a 32-bit Data Bus

Receive Descriptor 0 (RDES0)

RDES0 contains the received frame status, the frame length, and the descriptor ownership information. The descriptions in the following tables (**Table 1-83** through **Table 1-91**) are for the default mode of a Little-Endian 32-bit data bus with Same Endian descriptors, or Big-Endian data bus with Reverse-Endian descriptors. If the DUT is configured for Big-Endian mode, 32-bit data bus with Same Endian descriptors or Little-Endian data bus with Reverse Endian descriptors, then byte lanes 3 and 0 are swapped while byte lanes 1 and 2 are swapped, that is, bits [31:24] will be available on data bus [7:0], and vice-versa.

Table 30-23 Receive Descriptor 0

Bit	Description
31	**OWN: Own Bit** When set, this bit indicates that the descriptor is owned by the DMA of the GMAC Subsystem. When this bit is reset, this bit indicates that the descriptor is owned by the Host. The DMA clears this bit either when it completes the frame reception or when the buffers that are associated with this descriptor are full.
30	**AFM: Destination Address Filter Fail** When set, this bit indicates a frame that failed in the DA Filter in the GMAC Core.

Table 30-23 Receive Descriptor 0 (cont'd)

Bit	Description
29:16	**FL: Frame Length** These bits indicate the byte length of the received frame that was transferred to host memory (including CRC). This field is valid when Last Descriptor (RDES0[8]) is set and either the Descriptor Error (RDES0[14]) or Overflow Error bits are are reset. The frame length also includes the two bytes appended to the Ethernet frame when IP checksum calculation (Type 1) is enabled and the received frame is not a MAC control frame. This field is valid when Last Descriptor (RDES0[8]) is set. When the Last Descriptor and Error Summary bits are not set, this field indicates the accumulated number of bytes that have been transferred for the current frame.
15	**ES: Error Summary** Indicates the logical OR of the following bits: • RDES0[0]: Payload Checksum Error • RDES0[1]: CRC Error • RDES0[3]: Receive Error • RDES0[4]: Watchdog Timeout • RDES0[6]: Late Collision • RDES0[7]: IPC Checksum (Type 2) / Giant Frame • RDES0[11]: Overflow Error • RDES0[14]: Descriptor Error This field is valid only when the Last Descriptor (RDES0[8]) is set.
14	**DE: Descriptor Error** When set, this bit indicates a frame truncation caused by a frame that does not fit within the current descriptor buffers, and that the DMA does not own the Next Descriptor. The frame is truncated. This field is valid only when the Last Descriptor (RDES0[8]) is set.
13	**SAF: Source Address Filter Fail** When set, this bit indicates that the SA field of frame failed the SA Filter in the GMAC Core.
12	**LE: Length Error** When set, this bit indicates that the actual length of the frame received and that the Length/ Type field does not match. This bit is valid only when the Frame Type (RDES0[5]) bit is reset. Length error status is not valid when CRC error is present.
11	**OE: Overflow Error** When set, this bit indicates that the received frame was damaged due to buffer overflow in MTL.

Table 30-23 Receive Descriptor 0 (cont'd)

Bit	Description
10	**VLAN: VLAN Tag** When set, this bit indicates that the frame pointed to by this descriptor is a VLAN frame tagged by the GMAC Core.
9	**FS: First Descriptor** When set, this bit indicates that this descriptor contains the first buffer of the frame. If the size of the first buffer is 0, the second buffer contains the beginning of the frame. If the size of the second buffer is also 0, the next Descriptor contains the beginning of the frame.
8	**LS: Last Descriptor** When set, this bit indicates that the buffers pointed to by this descriptor are the last buffers of the frame
7	**IPC Checksum Error/Giant Frame** When IP Checksum Engine (Type 1) is enabled, this bit, when set, indicates that the 16-bit IPv4 Header checksum calculated by the core did not match the received checksum bytes. The Error Summary bit[15] is NOT set when this bit is set in this mode. If this bit is set when Full Checksum Offload Engine (Type 2) is enabled, it indicates an error in the IPv4 or IPv6 header. This error can be due to inconsistent Ethernet Type field and IP header Version field values, a header checksum mismatch in IPv4, or an Ethernet frame lacking the expected number of IP header bytes. Refer to **Table 1-84** for more details. If you do not select IP Checksum Module during core configuration, this bit, when set, indicates that the received frame was a Giant Frame. Giant frames are larger-than-1.518-byte (or 1.522-byte for VLAN) normal frames and larger-than-9,018-byte (9.022-byte for VLAN) frame when Jumbo Frame processing is enabled.
6	**LC: Late Collision** When set, this bit indicates that a late collision has occurred while receiving the frame in Half-Duplex mode.
5	**FT: Frame Type** When set, this bit indicates that the Receive Frame is an Ethernet-type frame (the LT field is greater than or equal to 16'h0600). When this bit is reset, it indicates that the received frame is an IEEE802.3 frame. This bit is not valid for Runt frames less than 14 bytes. Refer to **Table 1-84** for more details.
4	**RWT: Receive Watchdog Timeout** When set, this bit indicates that the Receive Watchdog Timer has expired while receiving the current frame and the current frame is truncated after the Watchdog Timeout.

Table 30-23 Receive Descriptor 0 (cont'd)

Bit	Description
3	**RE: Receive Error** When set, this bit indicates that the gmii_rxer_i signal is asserted while gmii_rxdv_i is asserted during frame reception. This error also includes carrier extension error in GMII and Half-duplex mode. Error can be of less/no extension, or error (rxd ≠ 0f) during extension.
2	**DE: Dribble Bit Error** When set, this bit indicates that the received frame has a non-integer multiple of bytes (odd nibbles). This bit is valid only in MII Mode.
1	**CE: CRC Error** When set, this bit indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received frame. This field is valid only when the Last Descriptor (RDES0[8]) is set.
0	**Rx MAC Address/Payload Checksum Error** When set, this bit indicates that the Rx MAC Address registers value (1 to 15) matched the frame's DA field. When reset, this bit indicates that the Rx MAC Address Register 0 value matched the DA field. If Full Checksum Offload Engine is enabled, this bit, when set, indicates the TCP, UDP, or ICMP checksum the core calculated does not match the received encapsulated TCP, UDP, or ICMP segment's Checksum field. This bit is also set when the received number of payload bytes does not match the value indicated in the Length field of the encapsulated IPv4 or IPv6 datagram in the received Ethernet frame. Refer to **Table 1-84** for more details.

When the Full Checksum Offload Engine (Type 2) is enabled, the permutations of bits 5, 7, and 0 reflect the conditions discussed in **Table 1-84**.

Table 30-24 Receive Descriptor 0 When COE (Type 2) Is Enabled

Bit 5: Frame Type	Bit 7: IPC Checksum Error	Bit 0: Payload Checksum Error	Frame Status
0	0	0	IEEE 802.3 Type frame (Length field value is less than 0600_H)
1	0	0	IPv4/IPv6 Type frame, no checksum error detected
1	0	1	IPv4/IPv6 Type frame with a payload checksum error (as described for PCE) detected

Table 30-24 Receive Descriptor 0 When COE (Type 2) Is Enabled (cont'd)

Bit 5: Frame Type	Bit 7: IPC Checksum Error	Bit 0: Payload Checksum Error	Frame Status
1	1	0	IPv4/IPv6 Type frame with an IP header checksum error (as described for IPC CE) detected
1	1	1	IPv4/IPv6 Type frame with both IP header and payload checksum errors detected
0	0	1	IPv4/IPv6 Type frame with no IP header checksum error and the payload check bypassed, due to an unsupported payload
0	1	1	A Type frame that is neither IPv4 or IPv6 (the Checksum Offload engine bypasses checksum completely.)
0	1	0	Reserved

Note: The first five conditions are backward-compatible to versions 3.30a and previous. The last two conditions (001, 011), which had been reserved, are not backward-compatible, because the Frame Type (FT) bit is reset during bypasses, even for valid Type frames.

Receive Descriptor 1 (RDES1)

RDES1 contains the buffer sizes and other bits that control the descriptor chain/ring.

*Note: See **Buffer Size Calculations** for further detail on calculating buffer sizes.*

Table 30-25 Receive Descriptor 1

Bit	Description
31	**Disable Interrupt on Completion** When set, this bit will prevent the setting of the RI (CSR5[6]) bit of the Status Register for the received frame that ends in the buffer pointed to by this descriptor. This, in turn, will disable the assertion of the interrupt to Host due to RI for that frame.
30:26	**Reserved**
25	**RER: Receive End of Ring** When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a Descriptor Ring.

Table 30-25 Receive Descriptor 1 (cont'd)

Bit	Description
24	**RCH: Second Address Chained** When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When RDES1[24] is set, RBS2 (RDES1[21-11]) is a "don't care" value. RDES1[25] takes precedence over RDES1[24].
23:22	**Reserved**
21:11	**RBS2: Receive Buffer 2 Size** These bits indicate the second data buffer size in bytes. The buffer size must be a multiple of 4/8/16 depending upon the bus widths (32/64/128), even if the value of RDES3 (buffer2 address pointer) is not aligned to bus width. In the case where the buffer size is not a multiple of 4/8/16, the resulting behavior is undefined. This field is not valid if RDES1[24] is set.
10:0	**RBS1: Receive Buffer 1 Size** Indicates the first data buffer size in bytes. The buffer size must be a multiple of 4/8/16 depending upon the bus widths (32/64/128), even if the value of RDES2 (buffer1 address pointer) is not aligned. In the case where the buffer size is not a multiple of 4/8/16, the resulting behavior is undefined. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or next descriptor depending on the value of RCH (Bit 24).

Receive Descriptor 2 (RDES2)

RDES2 contains the address pointer to the first data buffer in the descriptor.

*Note: See **Host Data Buffer Alignment** for further detail on buffer address alignment.*

Table 30-26 Receive Descriptor 2 (Default Operation)

Bit	Description
31:0	**Buffer 1 Address Pointer** These bits indicate the physical address of Buffer 1. There are no limitations on the buffer address alignment except for the following condition: The DMA uses the configured value for its address generation when the RDES2 value is used to store the start of frame. Note that the DMA performs a write operation with the RDES2[3/2/1:0] bits as 0 during the transfer of the start of frame but the frame data is shifted as per the actual Buffer address pointer. The DMA ignores RDES2[3/2/1:0] (corresponding to bus width of 128/64/32) if the address pointer is to a buffer where the middle or last part of the frame is stored.

Receive Descriptor 3 (RDES3)

RDES3 contains the address pointer either to the second data buffer in the descriptor or to the next descriptor.

Table 30-27 Receive Descriptor 3

Bit	Description
31:0	**Buffer 2 Address Pointer (Next Descriptor Address)** These bits indicate the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (RDES1[24]) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present. If RDES1[24] is set, the buffer (Next Descriptor) address pointer must be bus width-aligned (RDES3[3, 2, or 1:0] = 0, corresponding to a bus width of 128, 64, or 32. LSBs are ignored internally.) However, when RDES1[24] is reset, there are no limitations on the RDES3 value, except for the following condition: The DMA uses the configured value for its buffer address generation when the RDES3 value is used to store the start of frame. The DMA ignores RDES3[3, 2, or 1:0] (corresponding to a bus width of 128, 64, or 32) if the address pointer is to a buffer where the middle or last part of the frame is stored.

30.4.1.2 Transmit Descriptor

The descriptor addresses must be aligned to the bus width used (32). **Figure 30-26** shows the transmit descriptor format in Little-Endian mode with a 32-bit data bus.

Each descriptor is provided with two buffers, two byte-count buffers, and two address pointers, which enable the adapter port to be compatible with various types of memory-management schemes.

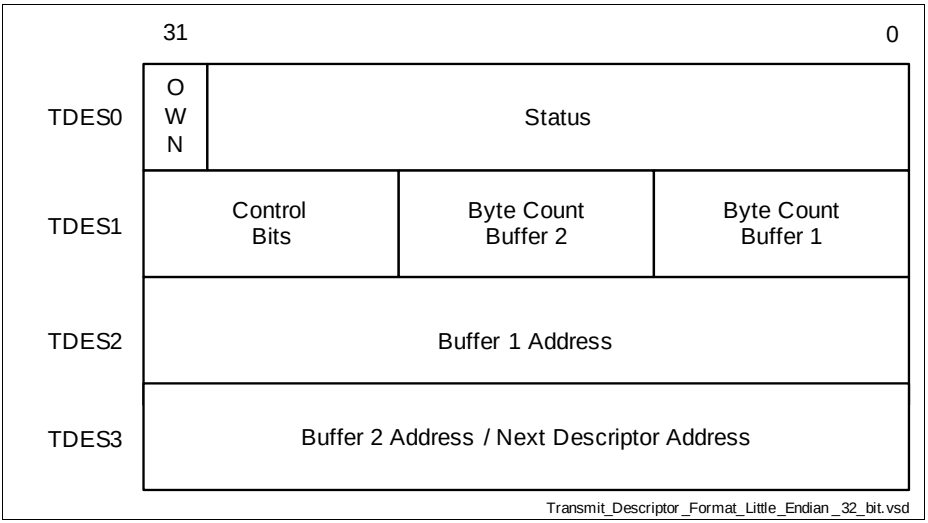


Figure 30-26 Transmit Descriptor Format in Little-Endian Mode With a 32-bit Data Bus

Transmit Descriptor 0 (TDES0)

TDES0 contains the transmitted frame status and the descriptor ownership information.

Table 30-28 Transmit Descriptor 0

Bit	Description
31	**OWN: Own Bit** When set, this bit indicates that the descriptor is owned by the DMA. When this bit is reset, this bit indicates that the descriptor is owned by the Host. The DMA clears this bit either when it completes the frame transmission or when the buffers allocated in the descriptor are empty. The ownership bit of the First Descriptor of the frame should be set after all subsequent descriptors belonging to the same frame have been set. This avoids a possible race condition between fetching a descriptor and the driver setting an ownership bit.
30:18	**Reserved**

Table 30-28 Transmit Descriptor 0 (cont'd)

Bit	Description
17	**TTSS: Tx Time Stamp Status** This status bit indicates that a time stamp has been captured for the corresponding transmit frame. When this bit is set, TDES2 and TDES3 have time stamp values that were captured for the transmit frame. This field is valid only when the Last Segment control bit (TDES1[30]) in a descriptor is set. This bit is valid only when IEEE1588 time stamping feature is enabled; otherwise, it is reserved.
16	**IHE: IP Header Error** When set, this bit indicates that the Checksum Offload engine detected an IP header error and consequently did not modify the transmitted frame for any checksum insertion. This bit is valid only when Full Checksum Offload is enabled; otherwise, it is reserved.
15	**ES: Error Summary** Indicates the logical OR of the following bits: • TDES0[14]: Jabber Timeout • TDES0[13]: Frame Flush • TDES0[11]: Loss of Carrier • TDES0[10]: No Carrier • TDES0[9]: Late Collision • TDES0[8]: Excessive Collision • TDES0[2]: Excessive Deferral • TDES0[1]: Underflow Error
14	**JT: Jabber Timeout** When set, this bit indicates the GMAC transmitter has experienced a jabber time-out. This bit is only set when the GMAC configuration register's JD bit is not set.
13	**FF: Frame Flushed** When set, this bit indicates that the DMA/MTL flushed the frame due to a SW flush command given by the CPU.

Table 30-28 Transmit Descriptor 0 (cont'd)

Bit	Description
12	**PCE: Payload Checksum Error** This bit, when set, indicates that the Checksum Offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure can be either due to insufficient bytes, as indicated by the IP Header's Payload Length field, or the MTL starting to forward the frame to the MAC transmitter in Store-and-Forward mode without the checksum having been calculated yet. This second error condition only occurs when the Transmit FIFO depth is less than the length of the Ethernet frame being transmitted: to avoid deadlock, the MTL starts forwarding the frame when the FIFO is full, even in Store-and-Forward mode. When the Full Checksum Offload engine is not enabled during configuration, this bit is reserved.
11	**LC: Loss of Carrier** When set, this bit indicates that Loss of Carrier occurred during frame transmission (that is, the gmii_crs_i signal was inactive for one or more transmit clock periods during frame transmission). This is valid only for the frames transmitted without collision and when the GMAC operates in Half-Duplex Mode.
10	**NC: No Carrier** When set, this bit indicates that the carrier sense signal form the PHY was not asserted during transmission.
9	**LC: Late Collision** When set, this bit indicates that frame transmission was aborted due to a collision occurring after the collision window (64 byte times including Preamble in MII Mode and 512 byte times including Preamble and Carrier Extension in GMII Mode). Not valid if Underflow Error is set.
8	**EC: Excessive Collision** When set, this bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current frame. If the DR (Disable Retry) bit in the GMAC Configuration Register is set, this bit is set after the first collision and the transmission of the frame is aborted.
7	**VF: VLAN Frame** When set, this bit indicates that the transmitted frame was a VLAN-type frame.
6:3	**CC: Collision Count** This 4-bit counter value indicates the number of collisions occurring before the frame was transmitted. The count is not valid when the Excessive Collisions bit (TDES0[8]) is set.

Table 30-28 Transmit Descriptor 0 (cont'd)

Bit	Description
2	**ED: Excessive Deferral** When set, this bit indicates that the transmission has ended because of excessive deferral of over 24,288 bit times (155,680 bits times in 1000 Mbit/s mode, or in Jumbo Frame enabled mode) if the Deferral Check (DC) bit is set high in the GMAC Control Register.
1	**UF: Underflow Error** When set, this bit indicates that the GMAC aborted the frame because data arrived late from the Host memory. Underflow Error indicates that the DMA encountered an empty Transmit Buffer while transmitting the frame. The transmission process enters the suspended state and sets both Transmit Underflow (Register 5[5]) and Transmit Interrupt (Register 5[0]).
0	**DB: Deferred Bit** When set, this bit indicates that the GMAC defers before transmission because of the presence of carrier. This bit is valid only in Half-Duplex mode.

Transmit Descriptor 1 (TDES1)

TDES1 contains the buffer sizes and other bits which control the descriptor chain/ring and the frame being transferred.

*Note: See **Buffer Size Calculations** for further detail on calculating buffer sizes.*

Table 30-29 Transmit Descriptor 1

Bit	Description
31	**IC: Interrupt on Completion** When set, this bit sets Transmit Interrupt (Register 5[0]) after the present frame has been transmitted.
30	**LS: Last Segment** When set, this bit indicates that the buffer contains the last segment of the frame.
29	**FS: First Segment** When set, this bit indicates that the buffer contains the first segment of a frame.

Table 30-29 Transmit Descriptor 1 (cont'd)

Bit	Description
28:27	**CIC: Checksum Insertion Control** These bits control the insertion of checksums in Ethernet frames that encapsulate TCP, UDP, or ICMP over IPv4 or IPv6 as described below. • 2'b00: Do nothing. Checksum Engine is bypassed • 2'b01: Insert IPv4 header checksum. Use this value to insert IPv4 header checksum when the frame encapsulates an IPv4 datagram. • 2'b10: Insert TCP/UDP/ICMP checksum. The checksum is calculated over the TCP, UDP, or ICMP segment only and the TCP, UDP, or ICMP pseudo-header checksum is assumed to be present in the corresponding input frame's Checksum field. An IPv4 header checksum is also inserted if the encapsulated datagram conforms to IPv4. • 2'b11: Insert a TCP/UDP/ICMP checksum that is fully calculated in this engine. In other words, the TCP, UDP, or ICMP pseudo-header is included in the checksum calculation, and the input frame's corresponding Checksum field has an all-zero value. An IPv4 Header checksum is also inserted if the encapsulated datagram conforms to IPv4. The Checksum engine detects whether the TCP, UDP, or ICMP segment is encapsulated in IPv4 or IPv6 and processes its data accordingly.
26	**DC: Disable CRC** When set, the GMAC does not append the Cyclic Redundancy Check (CRC) to the end of the transmitted frame. This is valid only when the first segment (TDES1[29]).
25	**TER: Transmit End of Ring** When set, this bit indicates that the descriptor list reached its final descriptor. The returns to the base address of the list, creating a descriptor ring.
24	**TCH: Second Address Chained** When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When TDES1[24] is set, TBS2 (TDES1[21–11]) are "don't care" values. TDES1[25] takes precedence over TDES1[24].
23	**DP: Disable Padding** When set, the GMAC does not automatically add padding to a frame shorter than 64 bytes. When this bit is reset, the DMA automatically adds padding and CRC to a frame shorter than 64 bytes and the CRC field is added despite the state of the DC (TDES1[26]) bit. This is valid only when the first segment (TDES1[29]) is set.

Table 30-29 Transmit Descriptor 1 (cont'd)

Bit	Description
22	**TTSE: Transmit Time Stamp Enable** When set, this bit enables IEEE1588 hardware time stamping for the transmit frame referenced by the descriptor. This field is valid only when the First Segment control bit (TDES1[29]) is set.
21:11	**TBS2: Transmit Buffer 2 Size** These bits indicate the Second Data Buffer in bytes. This field is not valid if TDES1[24] is set.
10:0	**TBS1: Transmit Buffer 1 Size** These bits indicate the First Data Buffer byte size. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or next descriptor depending on the value of TCH (Bit 24).

Transmit Descriptor 2 (TDES2)

TDES2 contains the address pointer to the first buffer of the descriptor.

Table 30-30 Transmit Descriptor 2

Bit	Description
31:0	**Buffer 1 Address Pointer** These bits indicate the physical address of Buffer 1. There is no limitation on the buffer address alignment. See **Host Data Buffer Alignment** for further detail on buffer address alignment.

Transmit Descriptor 3 (TDES3)

TDES3 contains the address pointer either to the second buffer of the descriptor or the next descriptor.

Table 30-31 Transmit Descriptor 3

Bit	Description
31:0	**Buffer 2 Address Pointer (Next Descriptor Address)** Indicates the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (TDES1[24]) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present. The buffer address pointer must be aligned to the bus width only when TDES1[24] is set. (LSBs are ignored internally.)

30.4.1.3 Descriptor Format With IEEE 1588 Time Stamping Enabled

The default descriptor format (as described in **"Receive Descriptor" on Page 1-677** and **"Transmit Descriptor" on Page 1-669**), and field descriptions remain unchanged when created by software (Own bit is set in DES0). However, if the software has enabled IEEE 1588 functionality, the DES2 and DES3 descriptor fields (see **Figure 30-27**) take on a different meaning when the DMA closes the descriptor (own bit in DES0 is cleared).

The DMA updates the DES2 and DES3 with the time stamp value before clearing the Own bit in DES0.

When the core is operating in 32-bit mode (**Figure 30-24**), DES2 is updated with the lower 32 time stamp bits (the Sub-Second field, called TSL in subsequent sections) and DES3 is updated with the upper 32 time stamp bits (the Seconds field, called TSH in subsequent sections).

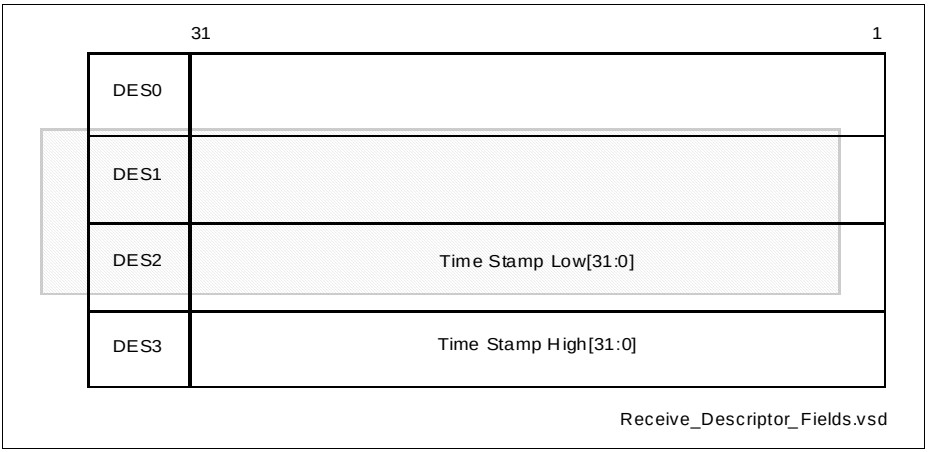


Figure 30-27 Receive Descriptor Fields When DMA Clears the Own Bit

The following sections describe the details specific to receive and transmit descriptors in this mode.

Receive Descriptor

Receive Time Stamp

The tables below describe the fields that have different meaning for RDES2 and RDES3 when the receive descriptor is closed and time stamping is enabled.

Note: When software disables the time stamping feature (the TSENA bit in Register 448 is low), the DMA does not update the descriptor's RDES2/RDES3 fields before closing the RDES0.

Table 30-32 Receive Descriptor Fields (RDES2)

Bit	Description
31:0	**RTSL: Receive Frame Time Stamp Low** The DMA updates this field with the least significant 32 bits of the time stamp captured for the corresponding receive frame. The DMA updates this field only for the last descriptor of the receive frame indicated by Last Descriptor status bit (RDES0[8]). When this field and the RTSH field in RDES3 show an all-ones value, the time stamp must be treated as corrupt.

Table 30-33 Receive Descriptor Fields (RDES3)

Bit	Description
31:0	**RTSH: Receive Frame Time Stamp High** The DMA updates this field with the most significant 32 bits of the time stamp captured for the corresponding receive frame. The DMA updates this field only for the last descriptor of the receive frame indicated by Last Descriptor status bit (RDES0[8]). When this field and RDES2's RTSL field show all-ones values, the time stamp must be treated as corrupt.

Transmit Descriptor

In addition to the changes described in **"Descriptor Format With IEEE 1588 Time Stamping Enabled" on Page 1-676**, the Transmit descriptor has additional control and status bits (TTSE and TTSS, respectively) for time stamping, as shown in **Figure 30-28**. Software sets the TTSE bit (when the Own bit is set), instructing the core to generate a time stamp for the corresponding Ethernet frame being transmitted. The DMA sets the TTSS bit if the time stamp has been updated in the TDES2 and TDES3 fields when the descriptor is closed (Own bit is cleared).

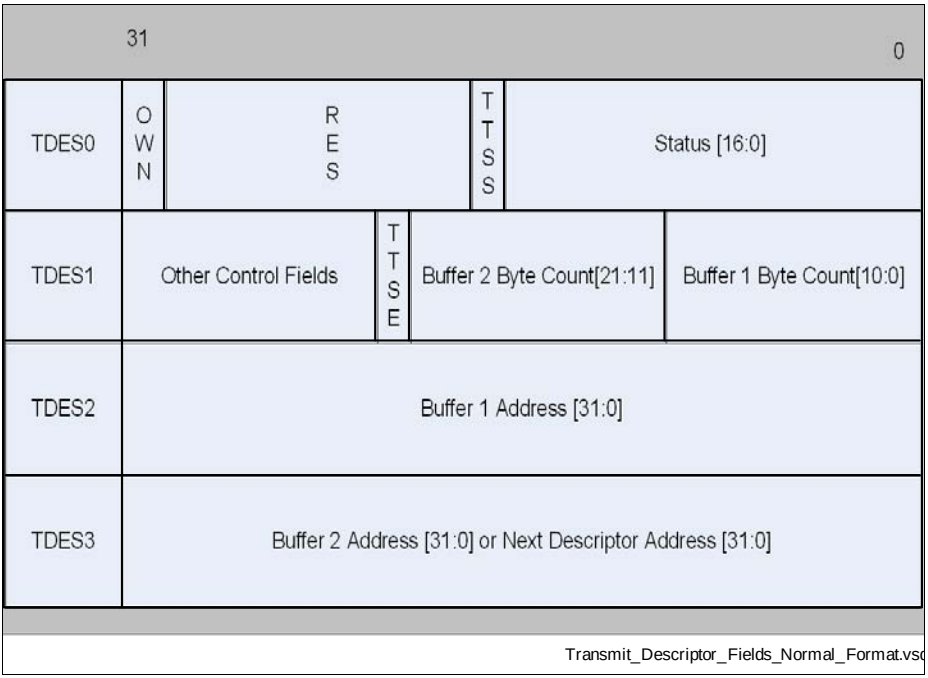


Figure 30-28 Transmit Descriptor Fields– Normal Format

Transmit Time Stamp Control and Status Fields

The position of these fields is different for normal transmit descriptor and enhanced format transmit descriptor. The value of this field in both the cases shall be preserved by the DMA at the time of closing the descriptor.

Updates to **Table 1-87** and **Table 1-88** for the default (normal) descriptor format are described below.

Table 30-34 Transmit Time Stamp Status – Normal Descriptor Format Case (TDES0)

Bit	Description
17	**TTSS: Transmit Time Stamp Status** This field is a status bit indicating that a time stamp was captured for the corresponding transmit frame. When this bit is set, both TDES2 and TDES3 have a time stamp value that was captured for the transmit frame. This field is valid only when the Last Segment control bit (TDES1[30] in the descriptor) is set.

Table 30-35 Transmit Time Stamp Control – Normal Descriptor Format Case (TDES1)

Bit	Description
22	**TTSE: Transmit Time Stamp Enable** When set, this field enables IEEE1588 hardware time stamping for the transmit frame described by the descriptor. This field is valid only when the First Segment control bit (TDES1[29] in the descriptor) is set.

Transmit Time Stamp Field

The transmit descriptor format and field descriptions remain unchanged when they are created by software (when the Own bit is set). However, when the DMA closes the last descriptor (marked, in the alternative descriptor format, by the LS bit in TDES1 or TDES0) and IEEE 1588 functionality is enabled (the Own bit is cleared), the TDES2 and TDES3 descriptor fields are updated with the time stamp, if taken, for that frame.

The fields in TDES2 and TDES3 are swapped when the core operates in 64- or 128-bit and with the descriptor in Reverse-Endian mode

Table 1-96 and **Table 1-97** describe the fields that have different meaning when the descriptor is closed.

Table 30-36 Transmit Descriptor Fields (TDES2)

Bit	Description
31:0	**TTSL: Transmit Frame Time Stamp Low** This field is updated by DMA with the least significant 32 bits of the time stamp captured for the corresponding transmit frame. This field has the time stamp only if the Last Segment control bit (LS) in the descriptor is set.

Table 30-37 Transmit Descriptor Fields (TDES3)

Bit	Description
31:0	**TTSH: Transmit Frame Time Stamp High** This field is updated by DMA with the most significant 32 bits of the time stamp captured for the corresponding transmit frame. This field has the time stamp only if the Last Segment control bit (LS) in the descriptor is set.

30.4.2 Alternate or Enhanced Descriptors

The alternate (or enhanced) descriptor structure can have 8 DWORDS (32-bytes) instead of the 4 DWORDS as in the case of normal descriptor format. The features of the alternate descriptor structure are

- The normal descriptor structure allows data buffers of up to 2.048 bytes. The alternative descriptor structure has been implemented to support buffers of up to 8 KB (useful for Jumbo frames).
- There is a re-assignment of control and status bits in TDES0, TDES1, RDES0 (Advanced time stamp or IPC full offload configuration), RDES1.
- The transmit descriptor stores the time stamp in TDES6 and TDES7 when advanced time stamp feature is selected.
- This receive descriptor structure is also used for storing the extended status (RDES4) and time stamp (RDES6 and RDES7) when advanced time stamp feature or IPC full offload is selected.
- When alternate descriptor mode is selected, and Time-stamping feature is enabled, the software needs to allocate 32-bytes (8 DWORDS) of memory for every descriptor. When Time-stamping or Receive IPC FullOffload engine are not enabled, the extended descriptors are not required and the SW can use alternate descriptors with the default size of 16 bytes. The core also needs to be configured for this change using the DMA Bus Mode Register[7].
- When alternate descriptor is chosen without Time Stamp or Full IPC Offload feature, the descriptor size is always 4 DWORDs (DES0-DES3).

The description or bit-mapping alternate descriptor structure (in Little Endian mode) is given below.

*Note: The effect of Big Endian mode (byte-swap) explained in **"Normal Descriptor Formats" on Page 1-658** apply to this descriptor structure as well.*
When alternate descriptor with only Full IPC Offload (Type 2) is selected, it is not backward compatible to the previous release 3.4x with respect to status bits[7,5,0] in RDES0. In this mode, you should enable the extended descriptor mode (8 DWORDS) to get the IPC checksum engine status in RDES4.

30.4.2.1 Transmit Descriptor

The transmit descriptor structure is shown in **Figure 30-29**. The application software must program the control bits TDES0[31:20] during descriptor initialization. When the DMA updates the descriptor, it write backs all the control bits except the OWN bit (which it clears) and updates the status bits[19:0]. The contents of the transmitter descriptor word 0 (TDES0) through word 3 (TDES3) are given in **Table 1-98** through **Table 1-101**, respectively.

With the advance time stamp support, the snapshot of the time stamp to be taken can be enabled for a given frame by setting the "TTSE: Transmit Time Stamp Enable" (bit-25 of TDES0). When the descriptor is closed (i.e. when the OWN bit is cleared), the time-

stamp is written into TDES6 and TDES7. This is indicated by the status bit “TTSS: Transmit Time Stamp Status” (bit-17 of TDES0). This is shown in **Figure 30-29**. The contents of TDES6 and TDES7 are mentioned in **Table 1-102** and **Table 1-103**.

Note: When either of Advanced Time Stamp or IPC Offload (Type 2) features is enabled, the SW should set the DMA Bus Mode register[7], so that the DMA operates with extended descriptor size. When this control bit is reset, the TDES4-TDES7 descriptor space are not valid.

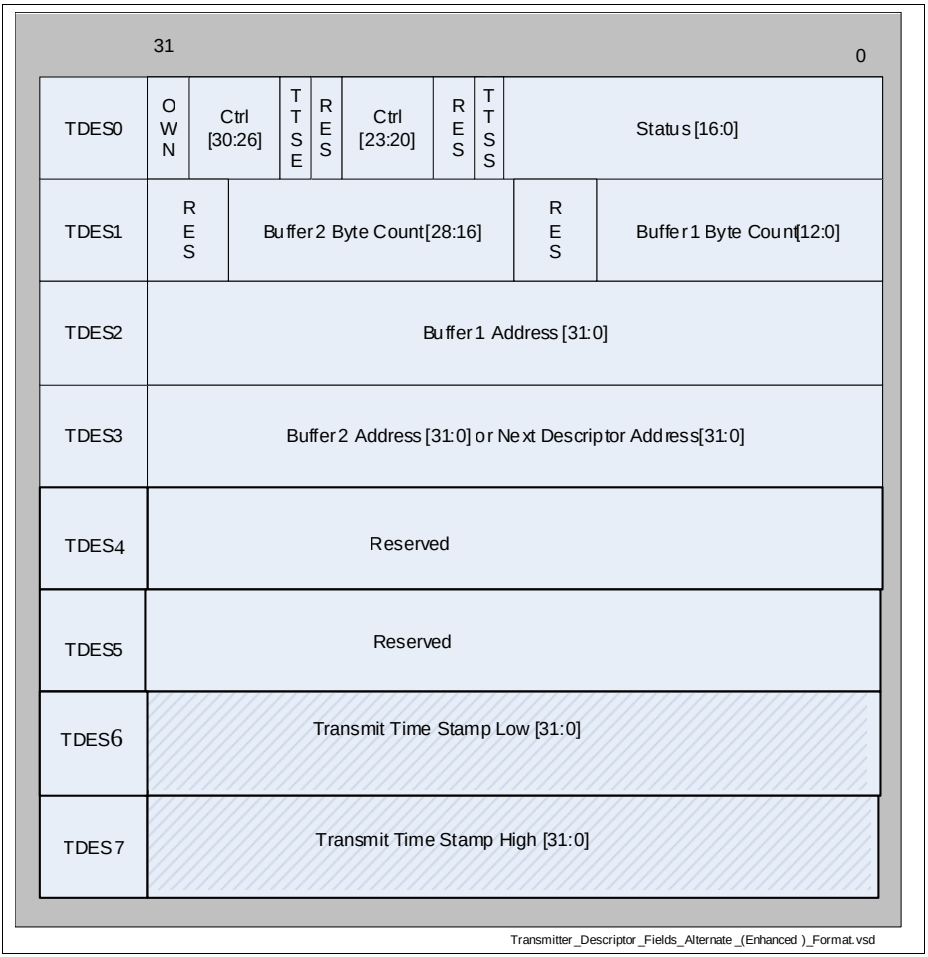


Figure 30-29 Transmitter Descriptor Fields - Alternate (Enhanced) Format

.

Table 30-38 Transmit Descriptor Word 0 (TDES0)

Bit	Description
31	**OWN: Own Bit** When set, this bit indicates that the descriptor is owned by the DMA. When this bit is reset, it indicates that the descriptor is owned by the Host. The DMA clears this bit either when it completes the frame transmission or when the buffers allocated in the descriptor are read completely. The ownership bit of the frame's first descriptor must be set after all subsequent descriptors belonging to the same frame have been set. This avoids a possible race condition between fetching a descriptor and the driver setting an ownership bit.
30	**IC: Interrupt on Completion** When set, this bit sets the Transmit Interrupt (Register 5[0]) after the present frame has been transmitted.
29	**LS: Last Segment** When set, this bit indicates that the buffer contains the last segment of the frame.
28	**FS: First Segment** When set, this bit indicates that the buffer contains the first segment of a frame.
27	**DC: Disable CRC** When this bit is set, the GMAC does not append a cyclic redundancy check (CRC) to the end of the transmitted frame. This is valid only when the first segment (TDES0[28]) is set.
26	**DP: Disable Pad** When set, the GMAC does not automatically add padding to a frame shorter than 64 bytes. When this bit is reset, the DMA automatically adds padding and CRC to a frame shorter than 64 bytes, and the CRC field is added despite the state of the DC (TDES0[27]) bit. This is valid only when the first segment (TDES0[28]) is set.
25	**TTSE: Transmit Time Stamp Enable** When set, this bit enables IEEE1588 hardware time stamping for the transmit frame referenced by the descriptor. This field is valid only when the First Segment control bit (TDES0[28]) is set.
24	**Reserved**

Table 30-38 Transmit Descriptor Word 0 (TDES0) (cont'd)

Bit	Description
23:22	**CIC: Checksum Insertion Control** These bits control the checksum calculation and insertion. Bit encodings are as shown below. • 2'b00: Checksum Insertion Disabled. • 2'b01: Only IP header checksum calculation and insertion are enabled. • 2'b10: IP header checksum and payload checksum calculation and insertion are enabled, but pseudo-header checksum is not calculated in hardware. • 2'b11: IP Header checksum and payload checksum calculation and insertion are enabled, and pseudo-header checksum is calculated in hardware. This field is reserved when the IPC_FULL_OFFLOAD configuration parameter is not selected.
21	**TER: Transmit End of Ring** When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a descriptor ring.
20	**TCH: Second Address Chained** When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When TDES0[20] is set, TBS2 (TDES1[28:16]) is a "don't care" value. TDES0[21] takes precedence over TDES0[20].
19:18	**Reserved**
17	**TTSS: Transmit Time Stamp Status** This field is used as a status bit to indicate that a time stamp was captured for the described transmit frame. When this bit is set, TDES2 and TDES3 have a time stamp value captured for the transmit frame. This field is only valid when the descriptor's Last Segment control bit (TDES0[29]) is set.
16	**IHE: IP Header Error** When set, this bit indicates that the GMAC transmitter detected an error in the IP datagram header. The transmitter checks the header length in the IPv4 packet against the number of header bytes received from the application and indicates an error status if there is a mismatch. For IPv6 frames, a header error is reported if the main header length is not 40 bytes. Furthermore, the Ethernet Length/Type field value for an IPv4 or IPv6 frame must match the IP header version received with the packet. For IPv4 frames, an error status is also indicated if the Header Length field has a value less than 0x5.

Table 30-38 Transmit Descriptor Word 0 (TDES0) (cont'd)

Bit	Description
15	**ES: Error Summary** Indicates the logical OR of the following bits: • TDES0[14]: Jabber Timeout • TDES0[13]: Frame Flush • TDES0[11]: Loss of Carrier • TDES0[10]: No Carrier • TDES0[9]: Late Collision • TDES0[8]: Excessive Collision • TDES0[2]: Excessive Deferral • TDES0[1]: Underflow Error • TDES0[16]: IP Header Error • TDES0[12]: IP Payload Error
14	**JT: Jabber Timeout** When set, this bit indicates the GMAC transmitter has experienced a jabber time-out. This bit is only set when the GMAC configuration register's JD bit is not set.
13	**FF: Frame Flushed** When set, this bit indicates that the DMA/MTL flushed the frame due to a software Flush command given by the CPU.
12	**IPE: IP Payload Error** When set, this bit indicates that GMAC transmitter detected an error in the TCP, UDP, or ICMP IP datagram payload. The transmitter checks the payload length received in the IPv4 or IPv6 header against the actual number of TCP, UDP, or ICMP packet bytes received from the application and issues an error status in case of a mismatch.
11	**LC: Loss of Carrier** When set, this bit indicates that a loss of carrier occurred during frame transmission (that is, the gmii_crs_i signal was inactive for one or more transmit clock periods during frame transmission). This is valid only for the frames transmitted without collision when the GMAC operates in Half-Duplex mode.
10	**NC: No Carrier** When set, this bit indicates that the Carrier Sense signal form the PHY was not asserted during transmission.
9	**LC: Late Collision** When set, this bit indicates that frame transmission was aborted due to a collision occurring after the collision window (64 byte-times, including preamble, in MII mode and 512 byte-times, including preamble and carrier extension, in GMII mode). This bit is not valid if the Underflow Error bit is set.

Table 30-38 Transmit Descriptor Word 0 (TDES0) (cont'd)

Bit	Description
8	**EC: Excessive Collision** When set, this bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current frame. If the DR (Disable Retry) bit in the GMAC Configuration register is set, this bit is set after the first collision, and the transmission of the frame is aborted.
7	**VF: VLAN Frame** When set, this bit indicates that the transmitted frame was a VLAN-type frame.
6:3	**CC: Collision Count** This 4-bit counter value indicates the number of collisions occurring before the frame was transmitted. The count is not valid when the Excessive Collisions bit (TDES0[8]) is set.
2	**ED: Excessive Deferral** When set, this bit indicates that the transmission has ended because of excessive deferral of over 24,288 bit times (155,680 bits times in 1,000-Mbit/s mode or if Jumbo Frame is enabled) if the Deferral Check (DC) bit in the GMAC Control register is set high.
1	**UF: Underflow Error** When set, this bit indicates that the GMAC aborted the frame because data arrived late from the Host memory. Underflow Error indicates that the DMA encountered an empty transmit buffer while transmitting the frame. The transmission process enters the Suspended state and sets both Transmit Underflow (Register 5[5]) and Transmit Interrupt (Register 5[0]).
0	**DB: Deferred Bit** When set, this bit indicates that the GMAC defers before transmission because of the presence of carrier. This bit is valid only in Half-Duplex mode.

Table 30-39 Transmit Descriptor Word 1 (TDES1)

Bit	Description
31:29	Reserved
28:16	**TBS2: Transmit Buffer 2 Size** These bits indicate the second data buffer size in bytes. This field is not valid if TDES0[20] is set. See **Buffer Size Calculations** for further detail on calculating buffer sizes.

Table 30-39 Transmit Descriptor Word 1 (TDES1) (cont'd)

Bit	Description
15:13	**Reserved**
12:0	**TBS1: Transmit Buffer 1 Size** These bits indicate the first data buffer byte size, in bytes. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or the next descriptor, depending on the value of TCH (TDES0[20]).

Table 30-40 Transmit Descriptor 2 (TDES2)

Bit	Description
31:0	**Buffer 1 Address Pointer** These bits indicate the physical address of Buffer 1. There is no limitation on the buffer address alignment. See **Host Data Buffer Alignment** for further detail on buffer address alignment.

Table 30-41 Transmit Descriptor 3 (TDES3)

Bit	Description
31:0	**Buffer 2 Address Pointer (Next Descriptor Address)** Indicates the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (TDES1[24]) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present. The buffer address pointer must be aligned to the bus width only when TDES1[24] is set. (LSBs are ignored internally.)

Table 30-42 Transmit Descriptor 6 (TDES6)

Bit	Description
31:0	**TTSL: Transmit Frame Time Stamp Low** This field is updated by DMA with the least significant 32 bits of the time stamp captured for the corresponding transmit frame. This field has the time stamp only if the Last Segment bit (LS) in the descriptor is set and Time stamp status (TTSS) bit is set.

Table 30-43 Transmit Descriptor 7 (TDES7)

Bit	Description
31:0	**TTSH: Transmit Frame Time Stamp High** This field is updated by DMA with the most significant 32 bits of the time stamp captured for the corresponding receive frame. This field has the time stamp only if the Last Segment bit (LS) in the descriptor is set and Time stamp status (TTSS) bit is set.

30.4.2.2 Receive Descriptor

The structure of the received descriptor is shown in **Figure 30-30**. This can have 32 bytes of descriptor data (8 DWORDs) when Advanced Time Stamping or IPC Full Offload feature is selected.

Note: When either of these features is enabled, the SW should set the DMA Bus Mode register[7] so that the DMA operates with extended descriptor size. When this control bit is reset, RDES0[7] and RDES0[0] will be always cleared and the RDES4-RDES7 descriptor space are not valid

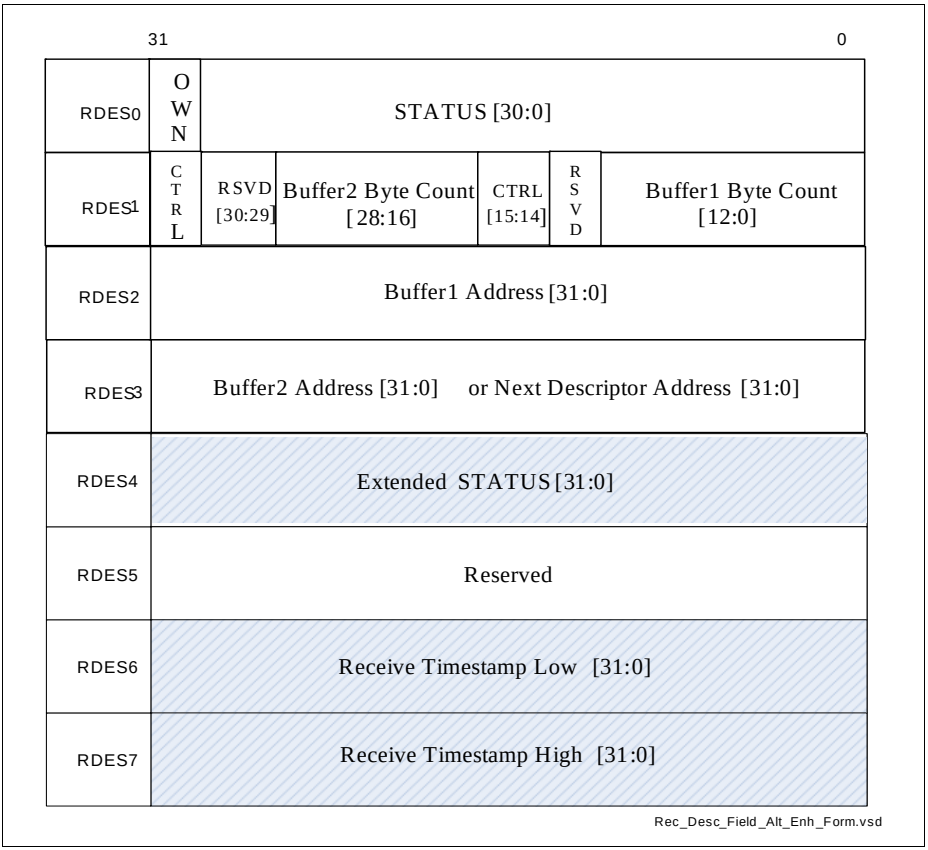


Figure 30-30 Receive Descriptor Fields - Alternate (Enhanced) Format

The contents of RDES0 are identified in **Table 1-104**. The contents of RDES1 through RDES3 are identified in **Table 1-105** through **Table 1-107**, respectively.

Note: Some of the bit functions of RDES0 are not backward compatible to Release 3.41a and previous versions. These bits are Bit[7], Bit[0] and Bit[5]. The function of Bit[5] is backward compatible to Release 3.30a and previous versions.

Table 30-44 Receive Descriptor Fields (RDES0)

Bit	Description
31	**OWN: Own Bit** When set, this bit indicates that the descriptor is owned by the DMA of the GMAC Subsystem. When this bit is reset, this bit indicates that the descriptor is owned by the Host. The DMA clears this bit either when it completes the frame reception or when the buffers that are associated with this descriptor are full.
30	**AFM: Destination Address Filter Fail** When set, this bit indicates a frame that failed in the DA Filter in the GMAC Core.
29:16	**FL: Frame Length** These bits indicate the byte length of the received frame that was transferred to host memory (including CRC). This field is valid when Last Descriptor (RDES0[8]) is set and either the Descriptor Error (RDES0[14]) or Overflow Error bits are are reset. The frame length also includes the two bytes appended to the Ethernet frame when IP checksum calculation (Type 1) is enabled and the received frame is not a MAC control frame. This field is valid when Last Descriptor (RDES0[8]) is set. When the Last Descriptor and Error Summary bits are not set, this field indicates the accumulated number of bytes that have been transferred for the current frame.
15	**ES: Error Summary** Indicates the logical OR of the following bits: • RDES0[1]: CRC Error • RDES0[3]: Receive Error • RDES0[4]: Watchdog Timeout • RDES0[6]: Late Collision • RDES0[7]: Giant Frame • RDES4[4:3]: IP Header/Payload Error • RDES0[11]: Overflow Error • RDES0[14]: Descriptor Error This field is valid only when the Last Descriptor (RDES0[8]) is set.
14	**DE: Descriptor Error** When set, this bit indicates a frame truncation caused by a frame that does not fit within the current descriptor buffers, and that the DMA does not own the Next Descriptor. The frame is truncated. This field is valid only when the Last Descriptor (RDES0[8]) is set.
13	**SAF: Source Address Filter Fail** When set, this bit indicates that the SA field of frame failed the SA Filter in the GMAC Core.

Table 30-44 Receive Descriptor Fields (RDES0) (cont'd)

Bit	Description
12	**LE: Length Error** When set, this bit indicates that the actual length of the frame received and that the Length/ Type field does not match. This bit is valid only when the Frame Type (RDES0[5]) bit is reset.
11	**OE: Overflow Error** When set, this bit indicates that the received frame was damaged due to buffer overflow in MTL.
10	**VLAN: VLAN Tag** When set, this bit indicates that the frame pointed to by this descriptor is a VLAN frame tagged by the GMAC Core.
9	**FS: First Descriptor** When set, this bit indicates that this descriptor contains the first buffer of the frame. If the size of the first buffer is 0, the second buffer contains the beginning of the frame. If the size of the second buffer is also 0, the next Descriptor contains the beginning of the frame.
8	**LS: Last Descriptor** When set, this bit indicates that the buffers pointed to by this descriptor are the last buffers of the frame
7	**Time Stamp Available/IP Checksum Error (Type1) /Giant Frame** When Advanced Time Stamp feature is present, this bit, when set, indicates that a snapshot of the timestamp is written in descriptor words 6 (RDES6) and 7 (RDES7). This is valid only when the Last Descriptor bit (RDES0[8]) is set. When IP Checksum Engine (Type 1) is selected, this bit, when set, indicates that the 16-bit IPv4 Header checksum calculated by the core did not match the received checksum bytes. Otherwise, this bit, when set, indicates the Giant Frame Status. Giant frames are larger-than-1,518-byte (or 1,522-byte for VLAN) normal frames and larger-than-9,018-byte (9,022-byte for VLAN) frame when Jumbo Frame processing is enabled.
6	**LC: Late Collision** When set, this bit indicates that a late collision has occurred while receiving the frame in Half-Duplex mode.
5	FT: Frame Type When set, this bit indicates that the Receive Frame is an Ethernet-type frame (the LT field is greater than or equal to 16'h0600). When this bit is reset, it indicates that the received frame is an IEEE802.3 frame. This bit is not valid for Runt frames less than 14 bytes.

Table 30-44 Receive Descriptor Fields (RDES0) (cont'd)

Bit	Description
4	**RWT: Receive Watchdog Timeout** When set, this bit indicates that the Receive Watchdog Timer has expired while receiving the current frame and the current frame is truncated after the Watchdog Timeout.
3	**RE: Receive Error** When set, this bit indicates that the gmii_rxer_i signal is asserted while gmii_rxdv_i is asserted during frame reception. This error also includes carrier extension error in GMII and Half-duplex mode. Error can be of less/no extension, or error (rxd ≠ 0f) during extension.
2	**DE: Dribble Bit Error** When set, this bit indicates that the received frame has a non-integer multiple of bytes (odd nibbles). This bit is valid only in MII Mode.
1	**CE: CRC Error** When set, this bit indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received frame. This field is valid only when the Last Descriptor (RDES0[8]) is set.
0	**Extended Status Available/Rx MAC Address** When either Advanced Time Stamp or IP Checksum Offload (Type 2) is present, this bit, when set, indicates that the extended status is available in descriptor word 4 (RDES4). This is valid only when the Last Descriptor bit (RDES0[8]) is set. When Advance Time Stamp Feature or IPC Full Offload is not selected, this bit indicates Rx MAC Address status. When set, this bit indicates that the Rx MAC Address registers value (1 to 31) matched the frame's DA field. When reset, this bit indicates that the Rx MAC Address Register 0 value matched the DA field.

Table 30-45 Receive Descriptor Fields 1 (RDES1)

Bit	Description
31	**DIC: Disable Interrupt on Completion** When set, this bit prevents setting the Status Register's RI bit (CSR5[6]) for the received frame ending in the buffer indicated by this descriptor. This, in turn, disables the assertion of the interrupt to Host due to RI for that frame.
30:29	**Reserved**

Table 30-45 Receive Descriptor Fields 1 (RDES1) (cont'd)

Bit	Description
28:16	**RBS2: Receive Buffer 2 Size** These bits indicate the second data buffer size, in bytes. The buffer size must be a multiple of 4, 8, or 16, depending on the bus widths (32, 64, or 128, respectively), even if the value of RDES3 (buffer2 address pointer) is not aligned to bus width. If the buffer size is not an appropriate multiple of 4, 8, or 16, the resulting behavior is undefined. This field is not valid if RDES1[14] is set. See **Buffer Size Calculations** for further details on calculating buffer sizes.
15	**RER: Receive End of Ring** When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a descriptor ring.
14	**RCH: Second Address Chained** When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When this bit is cleared, RBS2 (RDES1[28:16]) is a "don't care" value. RDES1[15] takes precedence over RDES1[14].
13	**Reserved**
12:0	**RBS1: Receive Buffer 1 Size** Indicates the first data buffer size in bytes. The buffer size must be a multiple of 4, 8, or 16, depending upon the bus widths (32, 64, or 128), even if the value of RDES2 (buffer1 address pointer) is not aligned. When the buffer size is not a multiple of 4, 8, or 16, the resulting behavior is undefined. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or next descriptor depending on the value of RCH (Bit 14). See **Buffer Size Calculations** for further details on calculating buffer sizes.

Table 30-46 Receive Descriptor Fields 2 (RDES2)

Bit	Description
31:0	**Buffer 1 Address Pointer** These bits indicate the physical address of Buffer 1. There are no limitations on the buffer address alignment except for the following condition: The DMA uses the configured value for its address generation when the RDES2 value is used to store the start of frame. Note that the DMA performs a write operation with the RDES2[3/2/1:0] bits as 0 during the transfer of the start of frame but the frame data is shifted as per the actual Buffer address pointer. The DMA ignores RDES2[3/2/1:0] (corresponding to bus width of 128/64/32) if the address pointer is to a buffer where the middle or last part of the frame is stored. See **Host Data Buffer Alignment** for further details on buffer address alignment.

Table 30-47 Receive Descriptor Fields 3 (RDES3)

Bit	Description
31:0	**Buffer 2 Address Pointer (Next Descriptor Address)** These bits indicate the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (RDES1[24]) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present. If RDES1[24] is set, the buffer (Next Descriptor) address pointer must be bus width-aligned (RDES3[3, 2, or 1:0] = 0, corresponding to a bus width of 128, 64, or 32. LSBs are ignored internally.) However, when RDES1[24] is reset, there are no limitations on the RDES3 value, except for the following condition: The DMA uses the configured value for its buffer address generation when the RDES3 value is used to store the start of frame. The DMA ignores RDES3[3, 2, or 1:0] (corresponding to a bus width of 128, 64, or 32) if the address pointer is to a buffer where the middle or last part of the frame is stored.

The extended status written is as shown in **Table 1-108**. The extended status is written only when there is status related to IPC or Time Stamp available. The availability of extended status is indicated by bit-0 of RDES0. This status is available only when Advance Time Stamp or IPC Full Offload feature is selected.

Table 30-48 Receive Descriptor Fields 4 (RDES4)

Bit	Description
31:14	**Reserved**
13	**PTP Version** When set, indicates that the received PTP message is having the IEEE 1588 version 2 format. When reset, it has the version 1 format. This is valid only if the message type is non-zero. This bit is available only if Advance Time Stamp feature is selected else it is reserved
12	**PTP Frame Type** When set, this bit that the PTP message is sent directly over Ethernet. When this bit is not set and the message type is non-zero, it indicates that the PTP message is sent over UDP-IPv4 or UDP-IPv6. The information on IPv4 or IPv6 can be obtained from bits 6 and 7. This bit is available only if Advanced Time Stamp feature is selected

Table 30-48 Receive Descriptor Fields 4 (RDES4) (cont'd)

Bit	Description
11:8	**Message Type** These bits are encoded to give the type of the message received. • 0000: No PTP message received • 0001: SYNC (all clock types) • 0010: Follow_Up (all clock types) • 0011: Delay_Req (all clock types) • 0100: Delay_Resp (all clock types) • 0101: Pdelay_Req (in peer-to-peer transparent clock) • 0110: Pdelay_Resp (in peer-to-peer transparent clock) • 0111: Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock) • 1000: Announce • 1001: Management • 1010: Signaling • 1011-1111: Reserved These bits are valid only when you select the Advance Time Stamp feature.
7	**IPv6 Packet Received** When set, this bit indicates that the received packet is an IPv6 packet.
6	**IPv4 Packet Received** When set, this bit indicates that the received packet is an IPv4 packet.
5	**IP Checksum Bypassed** When set, this bit indicates that the checksum offload engine is bypassed.
4	**IP Payload Error** When set, this bit indicates that the 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) that the core calculated does not match the corresponding checksum field in the received segment. It is also set when the TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field.

Table 30-48 Receive Descriptor Fields 4 (RDES4) (cont'd)

Bit	Description
3	**IP Header Error** When set, this bit indicates either that the 16-bit IPv4 header checksum calculated by the core does not match the received checksum bytes, or that the IP datagram version is not consistent with the Ethernet Type value.
2:0	**IP Payload Type** These bits indicate the type of payload encapsulated in the IP datagram processed by the Receive Checksum Offload Engine (COE). The COE also sets these bits to 2'b00 if it does not process the IP datagram's payload due to an IP header error or fragmented IP. • 3'b000: Unknown or did not process IP payload • 3'b001: UDP • 3'b010: TCP • 3'b011: ICMP • 3'b1xx: Reserved

RDES6 and RDES7 contain the snapshot of the time-stamp. The availability of the snapshot of the time-stamp in RDES6 and RDES7 is indicated by bit-7 in the RDES0 descriptor. The contents of RDES6 and RDES7 are identified in **Table 1-109** and **Table 1-110**, respectively.

Table 30-49 Receive Descriptor Fields 6 (RDES6)

Bit	Description
31:0	**RTSL: Receive Frame Time Stamp Low** This field is updated by DMA with the least significant 32 bits of the time stamp captured for the corresponding receive frame. This field is updated by DMA only for the last descriptor of the receive frame which is indicated by Last Descriptor status bit (RDES0[8]).

Table 30-50 Receive Descriptor Fields 7 (RDES7)

Bit	Description
31:0	**RTSH: Receive Frame Time Stamp High** This field is updated by DMA with the most significant 32 bits of the time stamp captured for the corresponding receive frame. This field is updated by DMA only for the last descriptor of the receive frame which is indicated by Last Descriptor status bit (RDES0[8]).

30.5 Ethernet MAC Module Implementation

This section describes the Ethernet MAC module interface as it is implemented in the TC21x/TC22x/TC23x. It especially covers clock control, port and on-chip connections, interrupt control, and address decoding.

30.5.1 Interface Connections of the Ethernet MAC Module

The Ethernet MAC module is supplied with a separate clock control, address decoding, and interrupt control logic. The modules' service request outputs are connected with one interrupt node.

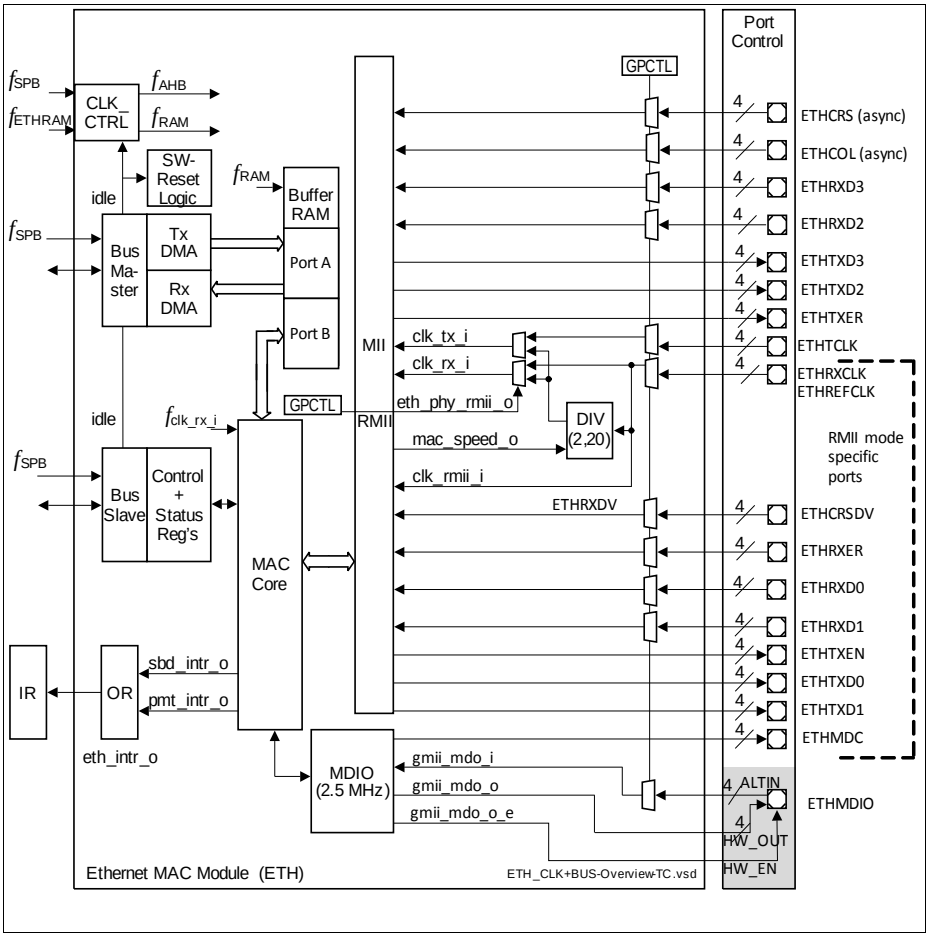


Figure 30-31 Ethernet MAC Module Implementation and Interconnections

30.5.1.1 On-Chip Connections

This section describes the on-chip connections of the Ethernet MAC module.

Interrupt Service Requests

The module has one Service Request Node connecting it to the interrupt system. The interrupt request line is connected to the interrupt controller as shown in **Table 1-112**.

Table 30-51 Service Request Lines of Ethernet MAC

Request Line	Connected to	Description
ETH_SRC0	eth_intr_o	DMA functions (sbd_intr_o), this internal line is connected via OR gate to ETH.SR0
ETH_SRC0	eth_intr_o	wake up on LAN (pmt_intr_o), this internal line is connected via OR gate to ETH.SR0

30.5.1.2 Clocks

The module has multiple clock inputs connecting it to the system. They are connected to the system as shown in **Table 1-113**.

Table 30-52 Clock Lines of Ethernet MAC

Clock Line	Connected to	Description
hclk_i / f_{AHB}	f_{SPB}	AHB Master Interface Clock, divided by 2 if GPCTL.DIV is set
clk_ram_i / f_{RAM}	f_{ETHRAM}	RAM Clock (separate clock from SCU.CGU) • it is by default 2 x f_{SPB} • must always be 2, 3, 4 ... (natural number) times higher than f_{SPB} • it must be ≥ 4 x clk_tx_i / clk_rx_i (Both are 25 MHz for 100 MBit/s and 2.5 MHz for 10 MBit/s) • divided by 2 if GPCTL.DIV is set
clk_csr_i	f_{SPB}	AHB Slave Interface Clock
clk_tx_i	ETHTXCLK (port pin)	Transmission Clock from Phy needs a clock input during SW reset (ETH_BUS_MODE.SWR)
clk_rx_i	ETHRXCLK (port pin)	Reception Clock from Phy needs a clock input during SW reset (ETH_BUS_MODE.SWR)

Table 30-52 Clock Lines of Ethernet MAC (cont'd)

Clock Line	Connected to	Description
clk_rmii_i	ETHREFCLK (port pin)	50-MHz clock used by the RMII from Phy needs a clock input during SW reset (ETH_BUS_MODE.SWR)
clk_ptp_ref_i	f_{SPB}	Reference Clock for the Time Stamp Update Logic

30.5.2 Ethernet MAC Module-Implementation Related Registers

The registers listed in **Figure 30-32** Ethernet MAC must be programmed for proper operation of the Ethernet MAC module.

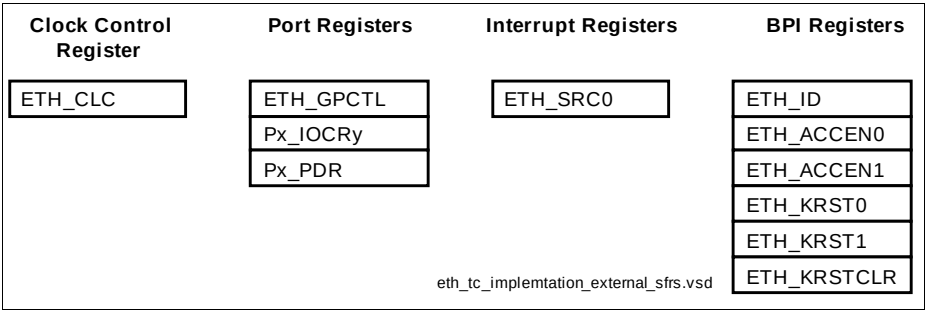


Figure 30-32 Ethernet MAC Implementation-specific Special Function Registers

30.5.2.1 Port Control

The Ethernet MAC is connected to general purpose I/O lines. Each line is connected to up to four different general purpose I/O lines that can be chosen alternatively. The selection from one of these alternatives is done by application SW by programming ETH_GPCTL.ALTIx respectively.

For MDIO, ALTIx selects the direction control signal (“HW_OUT control”) gmii_mdo_o_e_[x] as well. The signals gmii_mdo_o_e_[n] that are not selected are forced passive.

The following port control operations and selections must be executed for these I/O lines:

- Input/output function selection (Port IOCR registers)
- Pad driver characteristics selection for the outputs (Port PDR registers)

Table 30-53 shows an overview how bits and bit fields must be programmed for the required I/O functionality of the Ethernet MAC I/O lines.

Note: Some PHYs provide an interrupt output, signalling e.g. line activity. It might be useful to connect this to an interrupt input on system level. This signal is not connected to the Ethernet MAC*!*

Table 30-53 Ethernet MAC **I/O Control Selection and Setup**

Port	MAC signals	R/ Mii	Input Select Register	Input/Output Control Register Bits	I/ O
P02.8	ETHMDC		na	P02_IOCR8.PC8 = $1X110_B$	O

Table 30-53 Ethernet MAC **I/O Control Selection and Setup** (cont'd)

Port	MAC signals	R/ Mii	Input Select Register	Input/Output Control Register Bits	I/ O
P21.2	ETHMDC		na	P21_IOCR0.PC2 = $1X101_B$	O
P00.0	ETHMDIOA		ETH_GPCTL.ALTI0 = 00_B	P00_IOCR0.PC0 = $0XXX_B$	I/ O
	gmii_mdo_o_e_0 (I/O direction control of MDIO) connected to HW_DIR and HW_EN input of this port; choose Single EN - the pin is controlled by its own, dedicated, single HW_EN signal; gmii_mdo_o is connected to HW_OUT of the port, gmii_mdo_i is connected to ALTIN of the port (via input demultiplexer controlled by ALTI).				
P21.3	ETHMDIOD		ETH_GPCTL.ALTI0 = 11_B	P21_IOCR0.PC3 = $0XXX_B$	I/ O
	gmii_mdo_o_e_3 (I/O direction control of MDIO) connected to HW_DIR and HW_EN input of this port; choose Single EN - the pin is controlled by its own, dedicated, single HW_EN signal; gmii_mdo_o is connected to HW_OUT of the port, gmii_mdo_i is connected to ALTIN of the port (via input demultiplexer controlled by ALTI).				
P11.12	ETHRXCLKA / ETHREFCLKA	M / R	ETH_GPCTL.ALTI1 = 00_B	P11_IOCR12.PC12 = $0XXX_B$	I
	RMII: EXTCLK1 can be selected as output (P11_IOCR12.PC12 = $1X110_B$) while ETH_GPCTL.ALTI1 selects this pin as input. This setting supplies both: external PHY and this module with ETHREFCLK. Note that the jitter requirement of IEEE802.3 can not be met with this internal clock. If used as REFCLK, use fast input mode (P11_PDR1.PD12 = $X00_B$ and P11_IOCR12.PC12 = $0XXXX_B$)				
P11.12	ETHTXCLKB	R	ETH_GPCTL.ALTI10 =01_B	P11_IOCR12.PC12 = $0XXX_B$	I
	Not for application but for SW development and test. This allows to use MII loop back without external circuit if TXCLK and RXCLK are clocked with EXTCLK1.				
P11.11	ETHCRSB	M	ETH_GPCTL.ALTI2 = 01_B	P11_IOCR8.PC11 = $0XXX_B$	I
P10.1	ETHCRSC	M	ETH_GPCTL.ALTI2 = 10_B	P10_IOCR0.PC1 = $0XXX_B$	I
P10.2	ETHCOLB	M	ETH_GPCTL.ALTI3 = 01_B	P10_IOCR0.PC2 = $0XXX_B$	I

Table 30-53 Ethernet MAC **I/O Control Selection and Setup** (cont'd)

Port	MAC signals	R/ Mii	Input Select Register	Input/Output Control Register Bits	I/ O
P11.11	ETHRXDVA / ETHCRSDVA	M / R	ETH_GPCTL.ALTI4 = 00_B	P11_IOCR8.PC11 = $0XXX_B$	I
P21.7	ETHRXERB	R	ETH_GPCTL.ALTI5 = 01_B	P21_IOCR4.PC7 = $0XXX_B$	I
P10.3	ETHRXERC	R	ETH_GPCTL.ALTI5 = 10_B	P10_IOCR0.PC3 = $0XXX_B$	I
P11.10	ETHRXD0A	R	ETH_GPCTL.ALTI6 = 00_B	P11_IOCR8.PC10 = $0XXX_B$	I
P11.9	ETHRXD1A	R	ETH_GPCTL.ALTI7 = 00_B	P11_IOCR8.PC9 = $0XXX_B$	I
P11.8	ETHRXD2A	M	ETH_GPCTL.ALTI8 = 00_B	P11_IOCR8.PC8 = $0XXX_B$	I
P10.5	ETHRXD3B	M	ETH_GPCTL.ALTI9 = 01_B	P10_IOCR4.PC5 = $0XXX_B$	I
P13.1	ETHTXCLKC	M	ETH_GPCTL.ALTI10 =10_B	P13_IOCR0.PC1 = $0XXX_B$	I
P13.0	ETHTXER	M	na	P13_IOCR0.PC0 = $1X110_B$	O
P11.6	ETHTXEN	R	na	P11_IOCR4.PC6 = $1X110_B$	O
P11.3	ETHTXD0	R	na	P11_IOCR0.PC3 = $1X110_B$	O
P11.2	ETHTXD1	R	na	P11_IOCR0.PC2 = $1X110_B$	O
P13.3	ETHTXD2	M	na	P13_IOCR0.PC3 = $1X110_B$	O
P13.2	ETHTXD3	M	na	P13_IOCR0.PC2 = $1X110_B$	O

30.5.2.2 Clock Control

The master interface (DMAs) of the Ethernet MAC is connected to f_{SPB}via a clock control in register GPCTL.

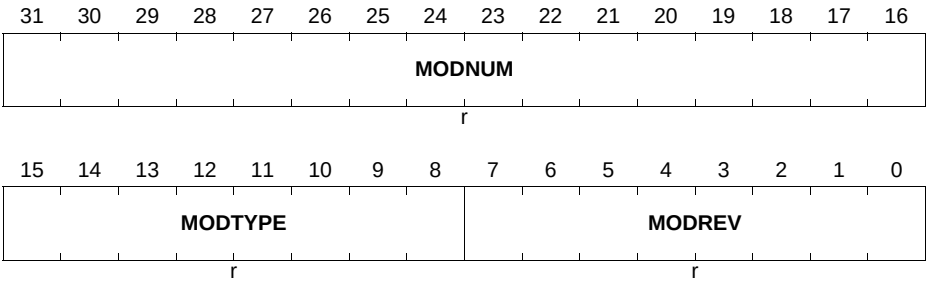
Note: The RAM frequency needs to run at double the application clock!

30.5.2.3 Additional Register

Module Identification Register

The Ethernet MAC Module Identification Register ID contains read-only information about the module version.

ETH_ID
Module Identification Register (0004$_H$) Reset Value: 00BF C0XX$_H$

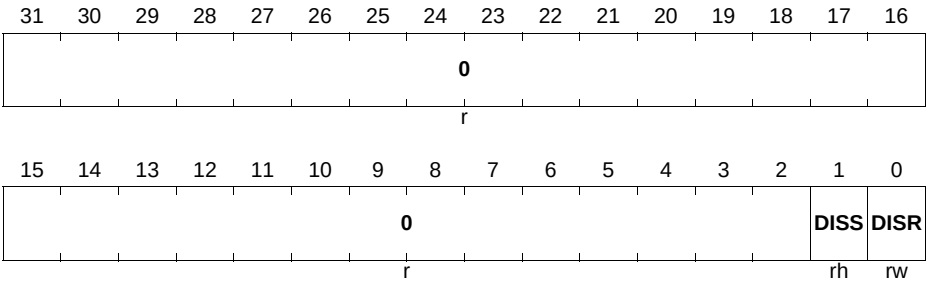


Field	Bits	Type	Description
MODREV	[7:0]	r	**Module Revision Number** This bit field defines the module revision number. The value of a module revision starts with 01$_H$ (first revision).
MODTYPE	[15:8]	r	**Module Type** This bit field defines the module as a 32-bit module: C0$_H$
MODNUM	[31:16]	r	**Module Number Value** This bit field defines the module identification number for the Ethernet MAC: 00BF$_H$

Clock Control Register (CLC)

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI_FPI for the module. Where a module kernel is connected to the CLC clock control interface, CLC controls the module clock for the module.

ETH_CLC
Clock Control Register **(0000_H)** **Reset Value: 0000 0003$_H$**



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. The disable request is delayed internally until all pending transactions on the master and slave interface to the SPB are completed.
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module.
0	[31:2]	r	**Reserved** Read as 0; should be written with 0.

Note: After a hardware reset operation, the $f_{Ethernet\ MAC}$ clock is switched off and the Ethernet MAC module is disabled (DISS set).

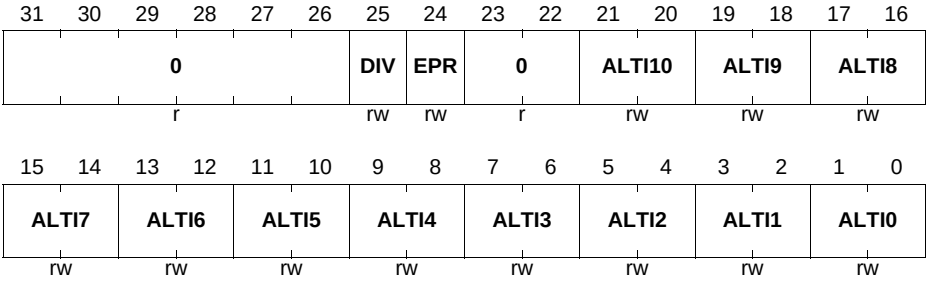
Input and Output Control Register Functions

The Input and Output Control Register GPCTLx determines for the Ethernet MAC which alternate input is to be used and selects the phy interface.

This register is reset by power on reset and hardware reset.

Note: This register is not affected by the local module reset!

ETH_GPCTL
Input and Output Control Register (0008_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
ALTIy **(y = 0-10)**	[2*y+1: 2*y]	rw	**Alternate Input Select** Selects the alternate input for channel y. If no signal is connected to an alternate input, it is connected to GND. 0000_B Alternate Input 0 selected 0001_B Alternate Input 1 selected $\ldots_B$ … 0011_B Alternate Input 3 selected

Field	Bits	Type	Description
EPR	24	rw	**External Phy Interface RMMI Mode Bit** Used to select the phy interface during module reset of MAC triggered by setting SWR bit of DMA Register 0. If set during this reset, RMII is selected and an internal predivider divides the clock clk_rmii_i from pin ETHRXC by 2 or 20 and provides it to the internal signals clk_tx_i and clk_rx_i. The division factor depends from the signal mac_speed_o, controlled by bit FES in Register 0 (MAC Configuration Register). If cleared during the module reset said above, MII is selected and the divider is not active but the clock signals clk_tx_i and clk_rx_i are connected directly to the referring port pins. Note that the status of this bit is latched in during module reset only.
DIV	25	rw	**Module Clock Divider Request Bit** If set, f_{AHB} and f_{RAM} will be divided by 2.
0	[31:26], [23:22]	r	**Reserved** Read as 0; should be written with 0.

Note: After a hardware reset, all clocks are switched off and the Ethernet module is disabled (DISS set). In this case, only register CLC is accessible for configuration.

Access Enable Register (ACCEN0)

The Access Enable Register 0 controls write access[1)] for transactions with the on chip bus master TAG ID 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000_B, EN1 -> TAG ID 000001_B, ... , EN31 -> TAG ID 011111_B.

1) The BPI_FPI Access Enable functionality controls only write transactions to the CLC, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers

ETH_ACCEN0

Access Enable Register 0 ($000C_H$) Reset Value: FFFF FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	EN30	EN29	EN28	EN27	EN26	EN25	EN24	EN23	EN22	EN21	EN20	EN19	EN18	EN17	EN16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n = 0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

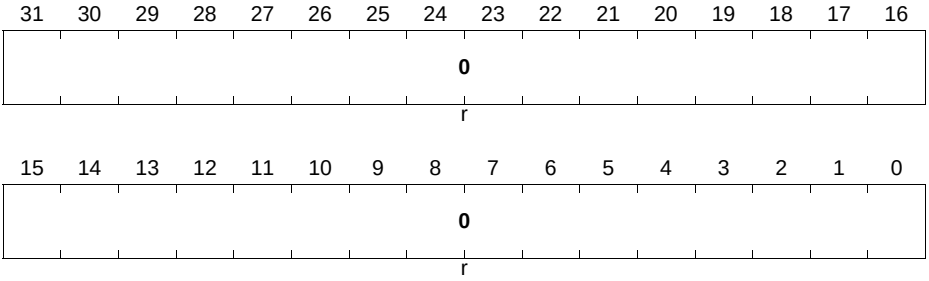
Access Enable Register (ACCEN1)

The Access Enable Register 1 controls write access[1)] for transactions with the on chip bus master TAG ID 100000_B to 111111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the AURIX devices.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000_B, EN1 -> TAG ID 100001_B, ... , EN31 -> TAG ID 111111_B.

ETH_ACCEN1

Access Enable Register 1 **(0010_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
0	[31:0]	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 0 (KRST0)

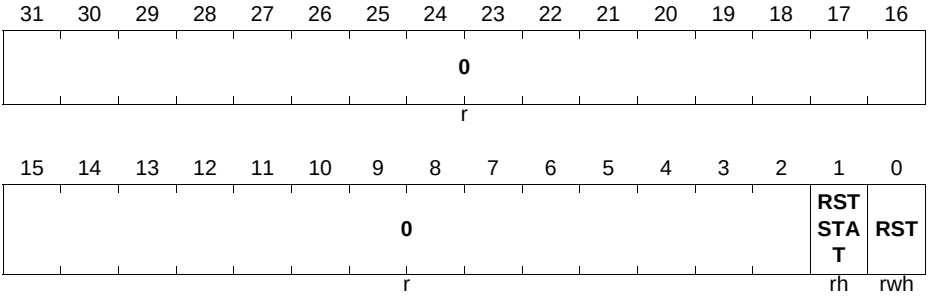
The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers.The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the KRSTCLE.CLR register bit.

Note: During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

ETH_KRST0

Kernel Reset Register 0 **(0014_H)** **Reset Value: 0000 0000_H**



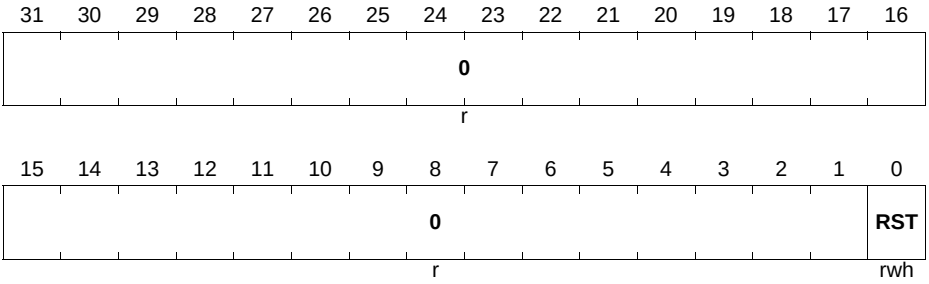
Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set and both: master and slave interface are idle. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed.
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. 0_B No kernel reset was executed 1_B Kernel reset was executed This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register.
0	[31:2]	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1 (KRST1)

The Kernel Reset Register 1 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is

necessary to set the RST bits by writing with ´1´ in both Kernel Reset registers. The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

ETH_KRST1
Kernel Reset Register 1 **(0018_H)** **Reset Value: $0000\ 0000_H$**

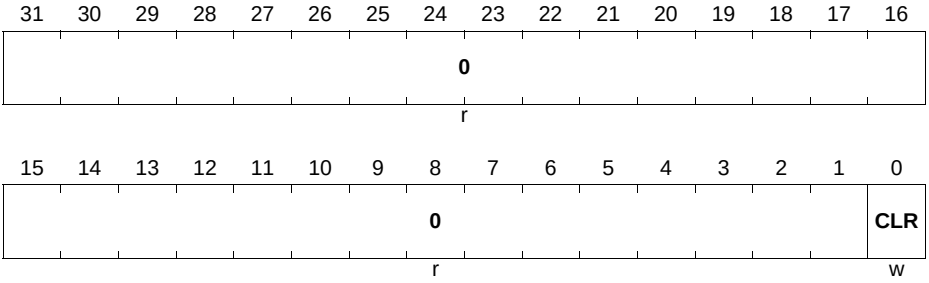


Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set and both: master and slave interface are idle. 0_B No kernel reset was requested 1_B A kernel reset was requested The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Status Clear Register (KRSTCLR)

The Kernel Reset Status Clear register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

ETH_KRSTCLR
Kernel Reset Status Clear Register (001C$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT Read always as 0.
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

30.6 Revision History

This User's Manual is based on: **Revision DWC Ether MAC 10/100/1000 Universal, V3.7a**".

Table 30-54 Revision History

Version Number	Changes to Previous Version
Rev_0.1	First Draft in IFX template
Rev_0.2	First configured version
Rev_0.3	First TS version
Rev_0.4	First ITS version
Rev_0.5	TS version
Rev_0.8	Removed unresolved cross references, Removed signal sbd_pwr_down_ack_o Added signals eth_phy_rmii_i and eth_rmii_speed_10_i
Rev_0.9	Updated Module Implementation Chapter
Rev_1.1	Expanded GMACREG19 headers for extraction
Rev_1.5	Removed non configured registers: GMACREG48 .. 54, DMAREG10 ..11 Updated reset values of registers GMACREG0, 8, 9, DMAREG22 Added clarification that normal descriptor format can not be supported
Rev_1.6	Updated Registers Chapter
Rev_1.7	Changed bit names to comply with compiler requirements - MAC_CONFIGURATION.2KPE -> TWOKPE - BUS_MODE.8xPBL -> PBLx8

Table 30-54 Revision History (cont'd)

Version Number	Changes to Previous Version
Rev_1.8	P21.1 ETHMDIO: No more support of I/O direction control by HW. P21.3: ETHMDIO mapping added Outputs remapped from O5 to O6 on all output ports P11.11 ETHCRSDV renamed to ETHCRSDVA P11.11 ETHCRSDVA: added ETHRXDVA and ETHCRSB P11.14 ETHCRS renamed to ETHCRSA P11.14 ETHCRSA: added ETHRXDVB and ETHCRSDVB P11.12: ETHTXCLKB mapping added P11.5 ETHTXCLK renamed ETHTXCLKA Fixed Synopsys issue: 9000522312 ("MAC inserts incorrect IP Payload Checksum at incorrect location for IPv6 packets with Authentication extension header") Fixed Synopsys issue: 9000522313 ("MAC provides incorrect IP Payload Checksum Error status when IPv6 packet with Authentication extension header is received") P21.2: ETHMDC added
Rev_1.9	Added port configuration hints for TXD0, TXD1, TXEN and REFCLK P11.13: ETHRXER renamed to ETHRXERA P21.7: ETHRXERB added
Rev_1.10	Port configuration changed for products with RMII only
Rev_1.11	Port configuration changed for TXD2 and TXD3 Added Port configuration for TC23x

w w w . i n f i n e o n . c o m

Published by Infineon Technologies AG