USER MANUAL

LPC2106/2105/2104 USER MANUAL

Preliminary
Supersedes data of 2003 Sep 17

2003 Oct 02

Philips
Semiconductors

PHILIPS

Table of Contents

List of Figures

List of Tables

DOCUMENT REVISION HISTORY

May, 2003:

- Prototype LPC2106/2105/2104 User Manual created from the design specification.

July, 2003:

- Flash Programming chapter added.
- Memory Accelerator Module chapter added.
- Register names in UARTs and timers updated.
- List of all registers added in the Introduction chapter.
- Pin Configuration chapter added.

August, 2003:

- MAM, VIC, GPIO, and RTC Usage Notes added.
- EmbeddedICE chapter updated.

September, 2003:

- Details on JTAG ports added in the EmbeddedICE chapter.
- Details on crystal oscillator added in the System Control Block chapter.
- List of possible baudrates when ISP is used added in the Flash Memory System and Programming chapter.
- Details on reset timing requirements added in the System Control Block, Reset chapter.

October, 2003:

- Number of Flash erase and write cycel is added in the Introduction chapter.

1. INTRODUCTION

FEATURES

- ARM7TDMI-S processor.
- 128 kilobyte on-chip Flash Program Memory with In-System Programming (ISP) and In-Application Programming (IAP) capability. Flash programming time is 1 ms for up to a 512 byte line. 10,000 erase and write cycles are guaranteed per 512 byte line. Single sector erase (8 kB) or the whole chip erase is done in 400 ms.
- 64/32/16 kilobyte Static RAM (LPC2106/2105/2104).
- Vectored Interrupt Controller.
- Emulation Trace Module supports real-time trace.
- RealMonitor module enables real time debugging.
- Standard ARM Test/Debug interface for compatibility with existing tools.
- Very small package LQFP48 ($7x7mm^2$).
- Two UARTs, one with full modem interface.
- I^2C serial interface.
- SPI serial interface.
- Two timers, each with 4 capture/compare channels.
- PWM unit with up to 6 PWM outputs.
- Real Time Clock.
- Watchdog Timer.
- General purpose I/O pins.
- CPU operating range up to 60 MHz.
- Dual power supply.
  - CPU operating voltage range of 1.65V to 1.95V (1.8V +/- 8.3%).
  - I/O power supply range of 3.0V to 3.6V (3.3V +/- 10%).
- Two low power modes, Idle and Power Down.
- Processor wakeup from Power Down mode via external interrupt.
- Individual enable/disable of peripheral functions for power optimization.
- On-chip crystal oscillator with an operating range of 10 MHz to 25 MHz.
- On-chip PLL allows CPU operation up to the maximum CPU rate. May be used over the entire crystal operating range.

APPLICATIONS

- Internet gateway.
- Serial communications protocol converter.
- Access control.
- Industrial Control.
- Medical equipment.

ARCHITECTURAL OVERVIEW

The LPC2106/2105/2104 consists of an ARM7TDMI-S CPU with emulation support, the ARM7 Local Bus for interface to on-chip memory controllers, the AMBA Advanced High-performance Bus (AHB) for interface to the interrupt controller, and the VLSI Peripheral Bus (VPB, a compatible superset of ARM's AMBA Advanced Peripheral Bus) for connection to on-chip peripheral functions. The LPC2106/2105/2104 configures the ARM7TDMI-S processor in little-endian byte order.

AHB peripherals are allocated a 2 megabyte range of addresses at the very top of the 4 gigabyte ARM memory space. Each AHB peripheral is allocated a 16 kilobyte address space within the AHB address space. LPC2106/05/04 peripheral functions (other than the interrupt controller) are connected to the VPB bus. The AHB to VPB bridge interfaces the VPB bus to the AHB bus. VPB peripherals are also allocated a 2 megabyte range of addresses, beginning at the 3.5 gigabyte address point. Each VPB peripheral is allocated a 16 kilobyte address space within the VPB address space.

The connection of on-chip peripherals to device pins is controlled by a Pin Connection Block. This must be configured by software to fit specific application requirements for the use of peripheral functions and pins.

ARM7TDMI-S PROCESSOR

The ARM7TDMI-S is a general purpose 32-bit microprocessor, which offers high performance and very low power consumption. The ARM architecture is based on Reduced Instruction Set Computer (RISC) principles, and the instruction set and related decode mechanism are much simpler than those of microprogrammed Complex Instruction Set Computers. This simplicity results in a high instruction throughput and impressive real-time interrupt response from a small and cost-effective processor core.

Pipeline techniques are employed so that all parts of the processing and memory systems can operate continuously. Typically, while one instruction is being executed, its successor is being decoded, and a third instruction is being fetched from memory.

The ARM7TDMI-S processor also employs a unique architectural strategy known as THUMB, which makes it ideally suited to high-volume applications with memory restrictions, or applications where code density is an issue.

The key idea behind THUMB is that of a super-reduced instruction set. Essentially, the ARM7TDMI-S processor has two instruction sets:

- The standard 32-bit ARM instruction set.
- A 16-bit THUMB instruction set.

The THUMB set's 16-bit instruction length allows it to approach twice the density of standard ARM code while retaining most of the ARM's performance advantage over a traditional 16-bit processor using 16-bit registers. This is possible because THUMB code operates on the same 32-bit register set as ARM code.

THUMB code is able to provide up to 65% of the code size of ARM, and 160% of the performance of an equivalent ARM processor connected to a 16-bit memory system.

The ARM7TDMI-S processor is described in detail in the ARM7TDMI-S Datasheet that can be found on official ARM website.

ON-CHIP FLASH MEMORY SYSTEM

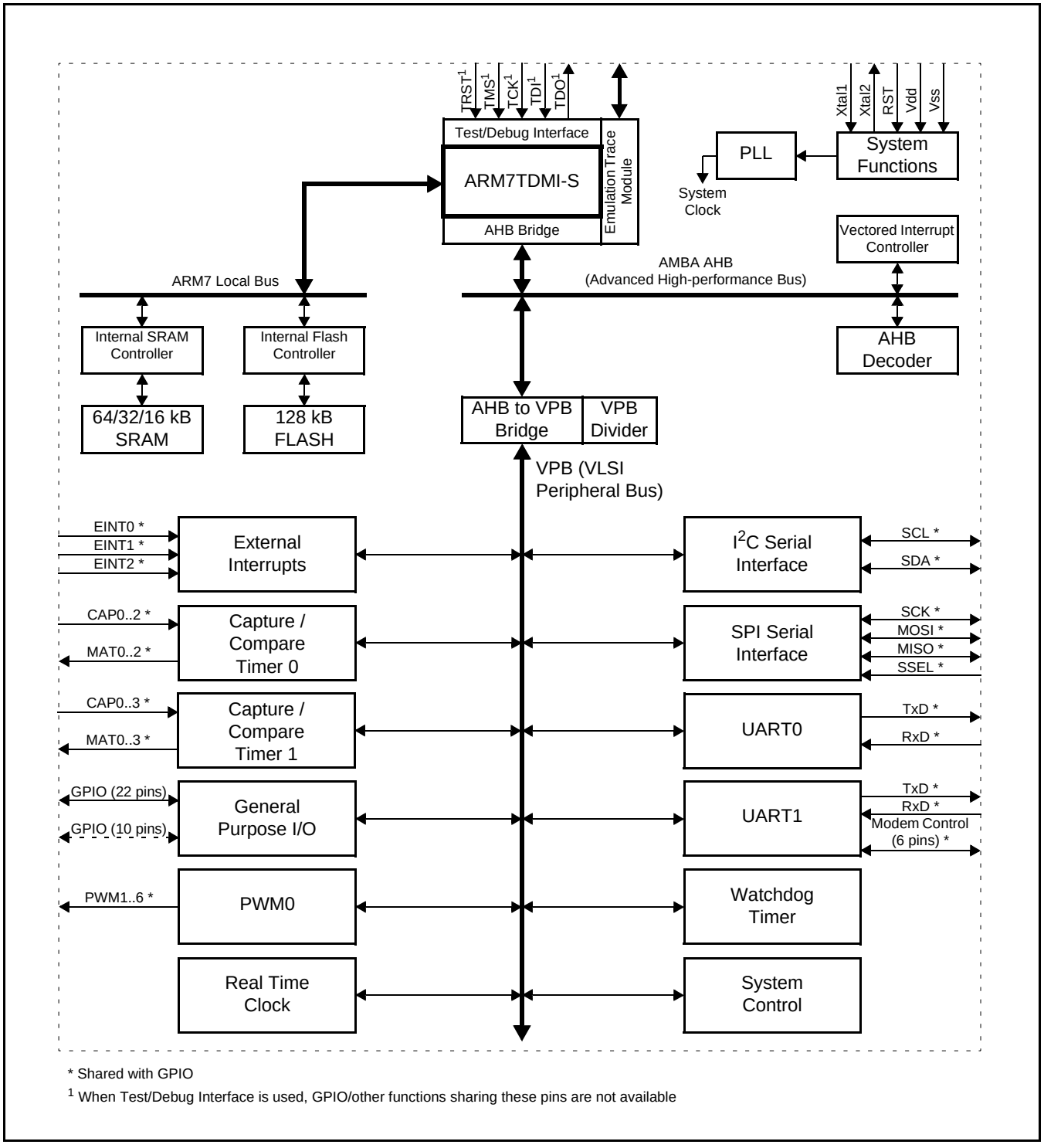
The LPC2106/2105/2104 incorporates a 128K byte Flash memory system. This memory may be used for both code and data storage. Programming of the Flash memory may be accomplished in several ways: over the serial built-in JTAG interface, using In System Programming (ISP) and UART0, or by means of In Application Programming (IAP) capabilities. The application program, using the In Application Programming (IAP) functions, may also erase and/or program the Flash while the application is running, allowing a great degree of flexibility for data storage field firmware upgrades, etc.

ON-CHIP STATIC RAM

The LPC2106, LPC2105 and LPC2104 provide a 64K byte, 32K byte and 16K byte static RAM memory respectively that may be used for code and/or data storage. The SRAM supports 8-bit, 16-bit, and 32-bit accesses.

The SRAM controller incorporates a write-back buffer in order to prevent CPU stalls during back-to-back writes. The write-back buffer always holds the last data sent by software to the SRAM. This data is only written to the SRAM when another write is requested by software. If a chip reset occurs, actual SRAM contents will not reflect the most recent write request. Any software that checks SRAM contents after reset must take this into account. A dummy write to an unused location may be appended to any operation in order to guarantee that all data has really been written into the SRAM.

BLOCK DIAGRAM



* Shared with GPIO

[1] When Test/Debug Interface is used, GPIO/other functions sharing these pins are not available

Figure 1: LPC2106/2105/2104 Block Diagram

LPC2106/2105/2104 REGISTERS

Accesses to registers in LPC2106/2105/2104 is restricted in the following ways:

1) user must NOT attempt to access any register locations not defined.

2) Access to any defined register locations must be strictly for the functions for the registers.

3) Register bits labeled '-', '0' or '1' can ONLY be written and read as follows:

- '-' MUST be written with '0', but can return any value when read (even if it was written with '0'). It is a reserved bit and may be used in future derivatives.
- '0' MUST be written with '0', and will return a '0' when read.
- '1' MUST be written with '1', and will return a '1' when read.

The following table shows all registers available in LPC2106/2105/2104 microcontroller sorted according to the address.

Access to the specific one can be categorized as either read/write, read only or write only (R/W, RO and WO respectively).

"Reset Value" field refers to the data stored in used/accessible bits only. It does not include reserved bits content. Some registers may contain undetermined data upon reset. In this case, reset value is categorized as "undefined". Classification as "NA" is used in case reset value is not applicable. Some registers in RTC are not affected by the chip reset. Their reset value is marked as * and these registers must be initialized by software if the RTC is enabled.

Registers in LPC2106/2105/2104 are 8, 16 or 32 bits wide. For 8 bit registers shown in Table 1, bit residing in the MSB (The Most Significant Bit) column corresponds to the bit 7 of that register, while bit in the LSB (The Least Significant Bit) column corresponds to the bit 0 of the same register.

If a register is 16/32 bit wide, the bit residing in the top left corner of its description, is the bit corresponding to the bit 15/31 of the register, while the bit in the bottom right corner corresponds to bit 0 of this register.

Examples: bit "ENA6" in PWMPCR register (address 0xE001404C) represents the bit at position 14 in this register; bits 15, 8, 7 and 0 in the same register are reserved. Bit "Stop on MR6" in PWMMCR register (0xE0014014) corresponds to the bit at position 20; bits 31 to 21 of the same register are reserved.

Unused (reserved) bits are marked with "-" and represented as gray fields. Access to them is restricted as already described.

Table 1: LPC2106/2105/2104 Registers

Address Offset	Name	Description	MSB							LSB	Access	Reset Value
WD												
0xE0000000	WD MOD	Watchdog mode register	-	-	-	-	WD INT	WD TOF	WDRE SET	WDEN	R/W	0
0xE0000004	WDTC	Watchdog timer constant register	32 bit data								R/W	0xFF
0xE0000008	WD FEED	Watchdog feed sequence register	8 bit data (0xAA fallowed by 0x55)								WO	NA

Table 1: LPC2106/2105/2104 Registers

Address Offset	Name	Description	MSB							LSB	Access	Reset Value
0xE000000C	WDTV	Watchdog timer value register	32 bit data								RO	0xFF
Timer 0												
0xE0004000	T0IR	T0 Interrupt Register	-	CR2 Int.	CR1 Int.	CR0 Int.	MR3 Int.	MR2 Int.	MR1 Int.	MR0 Int.	R/W	0
0xE0004004	T0TCR	T0 Control Register	-	-	-	-	-	-	CTR Enable	CTR Reset	R/W	0
0xE0004008	T0TC	T0 Counter	32 bit data								RW	0
0xE000400C	T0PR	T0 Prescale Register	32 bit data								R/W	0
0xE0004010	T0PC	T0 Prescale Counter	32 bit data								R/W	0
0xE0004014	T0MCR	T0 Match Control Register	4 reserved (-) bits				Stop on MR3	Reset on MR3	Int. on MR3	Stop on MR2	R/W	0
			Reset on MR2	Int. on MR2	Stop on MR1	Reset on MR1	Int. on MR1	Stop on MR0	Reset on MR0	Int. on MR0		
0xE0004018	T0MR0	T0 Match Register 0	32 bit data								R/W	0
0xE000401C	T0MR1	T0 Match Register 1	32 bit data								R/W	0
0xE0004020	T0MR2	T0 Match Register 2	32 bit data								R/W	0
0xE0004024	T0MR3	T0 Match Register 3	32 bit data								R/W	0
0xE0004028	T0CCR	T0 Capture Control Register	7 reserved (-) bits							Int. on Cpt.2	R/W	0
			Int. on Cpt.2 falling	Int. on Cpt.2 rising	Int. on Cpt.1	Int. on Cpt.1 falling	Int. on Cpt.1 rising	Int. on Cpt.0	Int. on Cpt.0 falling	Int. on Cpt.0 rising		
0xE000402C	T0CR0	T0 Capture Register 0	32 bit data								RO	0
0xE0004030	T0CR1	T0 Capture Register 1	32 bit data								RO	0
0xE0004034	T0CR2	T0 Capture Register 2	32 bit data								RO	0

Table 1: LPC2106/2105/2104 Registers

Address Offset	Name	Description	MSB							LSB	Access	Reset Value
0xE000403C	T0EMR	T0 External Match Register	6 reserved (-) bits						External Match Control 2		R/W	0
			External Match Control 1		External Match Control 0		-	Ext. Mtch2.	Ext. Mtch.1	Ext. Mtch.0		
Timer 1												
0xE0008000	T1IR	T1 Interrupt Register	CR3 Int.	CR2 Int.	CR1 Int.	CR0 Int.	MR3 Int.	MR2 Int.	MR1 Int.	MR0 Int.	R/W	0
0xE0008004	T1TCR	T1 Control Register	-	-	-	-	-	-	CTR Enable	CTR Reset	R/W	0
0xE0008008	T1TC	T1 Counter	32 bit data								RW	0
0xE000800C	T1PR	T1 Prescale Register	32 bit data								R/W	0
0xE0008010	T1PC	T1 Prescale Counter	32 bit data								R/W	0
0xE0008014	T1MCR	T1 Match Control Register	4 reserved (-) bits				Stop on MR3	Reset on MR3	Int. on MR3	Stop on MR2	R/W	0
			Reset on MR2	Int. on MR2	Stop on MR1	Reset on MR1	Int. on MR1	Stop on MR0	Reset on MR0	Int. on MR0		
0xE0008018	T1MR0	T1 Match Register 0	32 bit data								R/W	0
0xE000801C	T1MR1	T1 Match Register 1	32 bit data								R/W	0
0xE0008020	T1MR2	T1 Match Register 2	32 bit data								R/W	0
0xE0008024	T1MR3	T1 Match Register 3	32 bit data								R/W	0
0xE0008028	T1CCR	T1 Capture Control Register	4 reserved (-) bits				Int. on Cpt.3	Int. on Cpt.3 falling	Int. on Cpt.3 rising	Int. on Cpt.2	R/W	0
			Int. on Cpt.2 falling	Int. on Cpt.2 rising	Int. on Cpt.1	Int. on Cpt.1 falling	Int. on Cpt.1 rising	Int. on Cpt.0	Int. on Cpt.0 falling	Int. on Cpt.0 rising		
0xE000802C	T1CR0	T1 Capture Register 0	32 bit data								RO	0
0xE0008030	T1CR1	T1 Capture Register 1	32 bit data								RO	0
0xE0008034	T1CR2	T1 Capture Register 2	32 bit data								RO	0

Table 1: LPC2106/2105/2104 Registers

Address Offset	Name	Description	MSB							LSB	Access	Reset Value
0xE0008038	T1CR3	T1 Capture Register 3	32 bit data								RO	0
0xE000803C	T1EMR	T1 External Match Register	4 reserved (-) bits				External Match Control 3		External Match Control 2		R/W	0
			External Match Control 1		External Match Control 0		Ext. Mtch.3	Ext. Mtch2.	Ext. Mtch.1	Ext. Mtch.0		
UART 0												
0xE000C000	U0RBR (DLAB=0)	U0 Receiver Buffer Register	8 bit data								RO	un-defined
	U0THR (DLAB=0)	U0 Transmit Holding Register	8 bit data								WO	NA
	U0DLL (DLAB=1)	U0 Divisor Latch LSB	8 bit data								R/W	0x01
0xE000C004	U0IER (DLAB=0)	U0 Interrupt Enable Register	0	0	0	0	0	En. Rx Line Status Int.	Enable THRE Int.	En. Rx Data Av.Int.	R/W	0
	U0DLM (DLAB=1)	U0 Divisor Latch MSB	8 bit data								R/W	0
0xE000C008	U0IIR	U0 Interrupt ID Register	FIFOs Enabled		0	0	IIR3	IIR2	IIR1	IIR0	RO	0x01
	U0FCR	U0 FIFO Control Register	Rx Trigger		-	-	-	U0 Tx FIFO Reset	U0 Rx FIFO Reset	U0 FIFO Enable	WO	0
0xE000C00C	U0LCR	U0 Line Control Register	DLAB	Set Break	Stick Parity	Even Parity Select	Parity Enable	Nm. of Stop Bits	Word Length Select		R/W	0
0xE000C014	U0LSR	U0 Line Status Register	Rx FIFO Error	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60
0xE000C01C	U0SCR	U0 Scratch Pad Register	8 bit data								R/W	0
UART 1												

Table 1: LPC2106/2105/2104 Registers

Address Offset	Name	Description	MSB							LSB	Access	Reset Value
0xE0010000	U1RBR (DLAB=0)	U1 Receiver Buffer Register	8 bit data								RO	un-defined
	U1THR (DLAB=0)	U1 Transmit Holding Register	8 bit data								WO	NA
	U1DLL (DLAB=1)	U1 Divisor Latch LSB	8 bit data								R/W	0x01
0xE0010004	U1IER (DLAB=0)	U1 Interrupt Enable Register	0	0	0	0	En. Mdem Satus Int.	En. Rx Line Status Int.	Enable THRE Int.	En. Rx Data Av.Int.	R/W	0
	U1DLM (DLAB=1)	U1 Divisor Latch MSB	8 bit data								R/W	0
0xE0010008	U1IIR	U1 Interrupt ID Register	FIFOs Enabled		0	0	IIR3	IIR2	IIR1	IIR0	RO	0x01
	U1FCR	U1 FIFO Control Register	Rx Trigger		-	-	-	U0 Tx FIFO Reset	U0 Rx FIFO Reset	U0 FIFO Enable	WO	0
0xE001000C	U1LCR	U1 Line Control Register	DLAB	Set Break	Stick Parity	Even Parity Select	Parity Enable	Nm. of Stop Bits	Word Length Select		R/W	0
0xE0010010	U1 MCR	U1 Modem Control Register	0	0	0	Loop Back	0	0	RTS	DTR	R/W	0
0xE0010014	U1LSR	U1 Line Status Register	Rx FIFO Error	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60
0xE001001C	U1SCR	U1 Scratch Pad Register	8 bit data								R/W	0
0xE0010018	U1 MSR	U1 Modem Status Register	DCD	RI	DSR	CTS	Delta DCD	Trailing Edge RI	Delta DSR	Delta CTS	RO	0
PWM												
0xE0014000	PWM IR	PWM Interrupt Register	-	-	-	-	-	MR6 Int.	MR5 Int.	MR4 Int.	R/W	0
			-	-	-	-	MR3 Int.	MR2 Int.	MR1 Int.	MR0 Int.		
0xE0014004	PWM TCR	PWM Timer Control Register	-	-	-	-	PWM Enable	-	CTR Reset	CTR Enable	R/W	0
0xE0014008	PWM TC	PWM Timer Counter	32 bit data								RW	0

Table 1: LPC2106/2105/2104 Registers

Address Offset	Name	Description	MSB							LSB	Access	Reset Value
0xE001400C	PWM PR	PWM Prescale Register	32 bit data								R/W	0
0xE0014010	PWM PC	PWM Prescale Counter	32 bit data								R/W	0
0xE0014014	PWM MCR	PWM Match Control Register	11 reserved (-) bits			Stop on MR6	Reset on MR6	Int. on MR6	Stop on MR5	Reset on MR5	R/W	0
			Int. on MR5	Stop on MR4	Reset on MR4	Int. on MR4	Stop on MR3	Reset on MR3	Int. on MR3	Stop on MR2		
			Reset on MR2	Int. on MR2	Stop on MR1	Reset on MR1	Int. on MR1	Stop on MR0	Reset on MR0	Int. on MR0		
0xE0014018	PWM MR0	PWM Match Register 0	32 bit data								R/W	0
0xE001401C	PWM MR1	PWM Match Register 1	32 bit data								R/W	0
0xE0014020	PWM MR2	PWM Match Register 2	32 bit data								R/W	0
0xE0014024	PWM MR3	PWM Match Register 3	32 bit data								R/W	0
0xE0014040	PWM MR4	PWM Match Register 4	32 bit data								R/W	0
0xE0014044	PWM MR5	PWM Match Register 5	32 bit data								R/W	0
0xE0014048	PWM MR6	PWM Match Register 6	32 bit data								R/W	0
0xE001404C	PWM PCR	PWM Control Register	-	ENA6	ENA5	ENA4	ENA3	ENA2	ENA1	-	R/W	0
			-	SEL6	SEL5	SEL4	SEL3	SEL2	SEL1	-		
0xE0014050	PWM LER	PWM Latch Enable Register	-	Ena. PWM M6 Latch	Ena. PWM M5 Latch	Ena. PWM M4 Latch	Ena. PWM M3 Latch	Ena. PWM M2 Latch	Ena. PWM M1 Latch	Ena. PWM M0 Latch	R/W	0
I^2C												
0xE001C000	I2CON SET	I^2C Control Set Register	-	I2EN	STA	STO	SI	AA	-	-	R/W	0
0xE001C004	I2STAT	I^2C Status Register	5 bit Status					0	0	0	RO	0xF8
0xE001C008	I2DAT	I^2C Data Register	8 bit data								R/W	0

Table 1: LPC2106/2105/2104 Registers

Address Offset	Name	Description	MSB							LSB	Access	Reset Value
0xE001C00C	I2 ADR	I^2C Slave Address Register	7 bit data							GC	R/W	0
0xE001C010	I2 SCLH	SCL Duty Cycle Register High Half Word	16 bit data								R/W	0x04
0xE001C014	I2 SCLL	SCL Duty Cycle Register Low Half Word	16 bit data								R/W	0x04
0xE001C018	I2CON CLR	I^2C Control Clear Register	-	I2ENC	STAC	-	SIC	AAC	-	-	WO	NA
SPI												
0xE0020000	SPCR	SPI Control Register	SPIE	LSBF	MSTR	CPOL	CPHA	-	-	-	R/W	0
0xE0020004	SPSR	SPI Status Register	SPIF	WCOL	ROVR	MODF	ABRT	-	-	-	RO	0
0xE0020008	SPDR	SPI Data Register	8 bit data								R/W	0
0xE002000C	SP CCR	SPI Clock Counter Register	8 bit data								R/W	0
0xE002001C	SPINT	SPI Interrupt Flag	-	-	-	-	-	-	-	SPI Int.	R/W	0
RTC												
0xE0024000	ILR	Interrupt Location Register	-	-	-	-	-	-	RTC ALF	RTC CIF	R/W	*
0xE0024004	CTC	Clock Tick Counter	15 bit data							-	RO	*
0xE0024008	CCR	Clock Control Register	-	-	-	-	CTTEST		CTC RST	CLK EN	R/W	*
0xE002400C	CIIR	Counter Increment Interrupt Register	IM YEAR	IM MON	IM DOY	IM DOW	IM DOM	IM HOUR	IM MIN	IM SEC	R/W	*
0xE0024010	AMR	Alarm Mask Register	AMR YEAR	AMR MON	AMR DOY	AMR DOW	AMR DOM	AMR HOUR	AMR MIN	AMR SEC	R/W	*

Table 1: LPC2106/2105/2104 Registers

Address Offset	Name	Description	MSB							LSB	Access	Reset Value
0xE0024014	CTIME0	Consolidated Time Register 0	-	-	-	-	-	3 bit Day of Week			RO	*
			-	-	-	5 bit Hours						
			-	-	6 bit Minutes							
			-	-	6 bit Seconds							
0xE0024018	CTIME1	Consolidated Time Register 1	-	-	-	-	12 bit Year				RO	*
			-	-	-	-	4 bit Month					
			-	-	-	5 bit Day of Month						
0xE002401C	CTIME2	Consolidated Time Register 2	reserved (-) 20 bits				12 bit Day of Year				RO	*
0xE0024020	SEC	Seconds Register	-	-	6 bit data						R/W	*
0xE0024024	MIN	Minutes Register	-	-	6 bit data						R/W	*
0xE0024028	HOUR	Hours Register	-	-	-	5 bit data					R/W	*
0xE002402C	DOM	Day of Month Register	-	-	-	5 bit data					R/W	*
0xE0024030	DOW	Day of Week Register	-	-	-	-	-	3 bit data			R/W	*
0xE0024034	DOY	Day of Year Register	reserved (-) 7 bits				9 bit data				R/W	*
0xE0024038	MONTH	Months Register	-	-	-	-	4 bit data				R/W	*
0xE002403C	YEAR	Year Register	reserved (-) 4 bits			12 bit data					R/W	*
0xE0024060	AL SEC	Alarm value for Seconds	-	-	6 bit data						R/W	*
0xE0024064	AL MIN	Alarm value for Minutes	-	-	6 bit data						R/W	*
0xE0024068	AL HOUR	Alarm value for Hours	-	-	-	5 bit data					R/W	*
0xE002406C	AL DOM	Alarm value for Day of Month	-	-	-	5 bit data					R/W	*
0xE0024070	AL DOW	Alarm value for Day of Week	-	-	-	-	-	3 bit data			R/W	*

Table 1: LPC2106/2105/2104 Registers

Address Offset	Name	Description	MSB							LSB	Access	Reset Value
0xE0024074	AL DOY	Alarm value for Day of Year	reserved (-) 7 bits			9 bit data					R/W	*
0xE0024078	AL MON	Alarm value for Months	-	-	-	-	4 bit data				R/W	*
0xE002407C	AL YEAR	Alarm value for Year	reserved (-) 4 bits			12 bit data					R/W	*
0xE0024080	PRE INT	Prescale value, integer portion	reserved (-) 3 bits			13 bit data					R/W	0
0xE0024084	PRE FRAC	Prescale value, fractional portion	-	15 bit data							R/W	0
GPIO												
0xE0028000	IOPIN	GPIO Pin value register	32 bit data								RO	NA
0xE0028004	IOSET	GPIO 0 Output set register	32 bit data								R/W	0
0xE0028008	IODIR	GPIO 0 Direction control register	32 bit data								R/W	0
0xE002800C	IOCLR	GPIO 0 Output clear register	32 bit data								WO	0
Pin Connet Block												
0xE002C000	PIN SEL0	Pin function select register 0	32 bit data								R/W	0
0xE002C004	PIN SEL1	Pin function select register 1	32 bit data								R/W	0
System Control Block												
0xE01FC000	MAM CR	MAM control register	-	-	-	-	-	-	2 bit data		R/W	0
0xE01FC004	MAM TIM	MAM timing control	-	-	-	-	-	3 bit data			R/W	0x07

Table 1: LPC2106/2105/2104 Registers

Address Offset	Name	Description	MSB							LSB	Access	Reset Value
0xE01FC040	MEM MAP	Memory mapping control	-	-	-	-	-	-	2 bit data		R/W	0
0xE01FC080	PLL CON	PLL control register	-	-	-	-	-	-	PLLC	PLLE	R/W	0
0xE01FC084	PLL CFG	PLL configuration register	-	2bit data PSEL		5 bit data MSEL					R/W	0
0xE01FC088	PLL STAT	PLL status register	-	-	-	-	-	PLOCK	PLLC	PLLE	RO	0
			-	2bit data PSEL		5 bit data MSEL						
0xE01FC08C	PLL FEED	PLL feed register	8 bit data								WO	NA
0xE01FC0C0	PCON	Power control register	-	-	-	-	-	-	PD	IDL	R/W	0
0xE01FC0C4	PCONP	Power control for peripherals	reserved (-) 22 bits						PC RTC	PC SPI	R/W	0x3BE
			PC I2C	-	PC PWM0	PC URT1	PC URT0	PC TIM1	PC TIM0	-		
0xE01FC100	VPB DIV	VPB divider control	-	-	-	-	-	-	2 bit data		R/W	0
0xE01FC140	EXT INT	External interrupt flag register	-	-	-	-	-	EINT2	EINT1	EINT0	R/W	0
0xE01FC144	EXT WAKE	External interrupt wakeup register	-	-	-	-	-	EXT WAKE 2	EXT WAKE 1	EXT WAKE 0	R/W	0

2. LPC2106/2105/2104 MEMORY ADDRESSING

MEMORY MAPS

The LPC2106/2105/2104 incorporates several distinct memory regions, shown in the following figures. Figure 2 shows the overall map of the entire address space from the user program viewpoint following reset. The interrupt vector area supports address re-mapping, which is described later in this section.

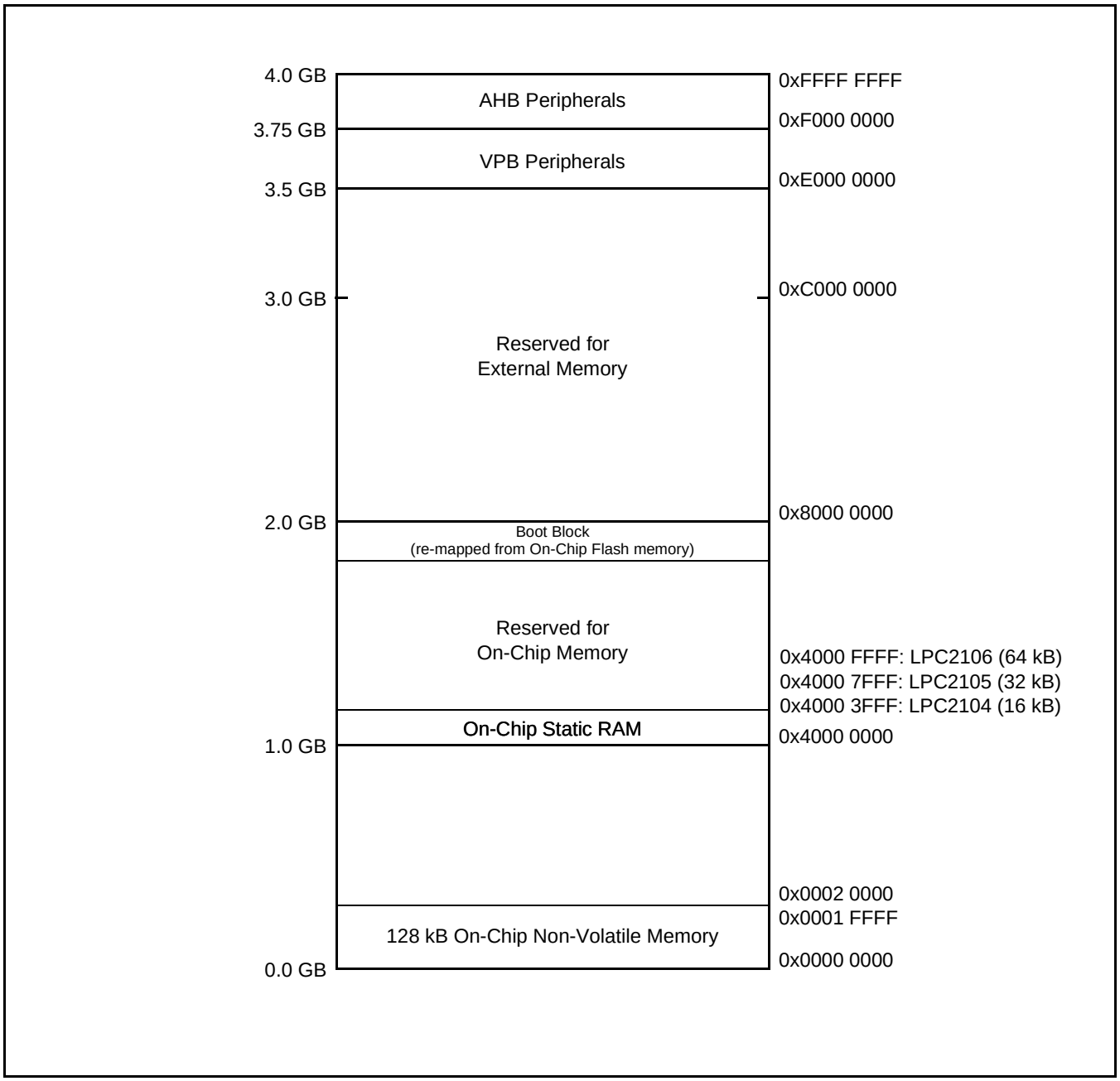


Figure 2: System Memory Map

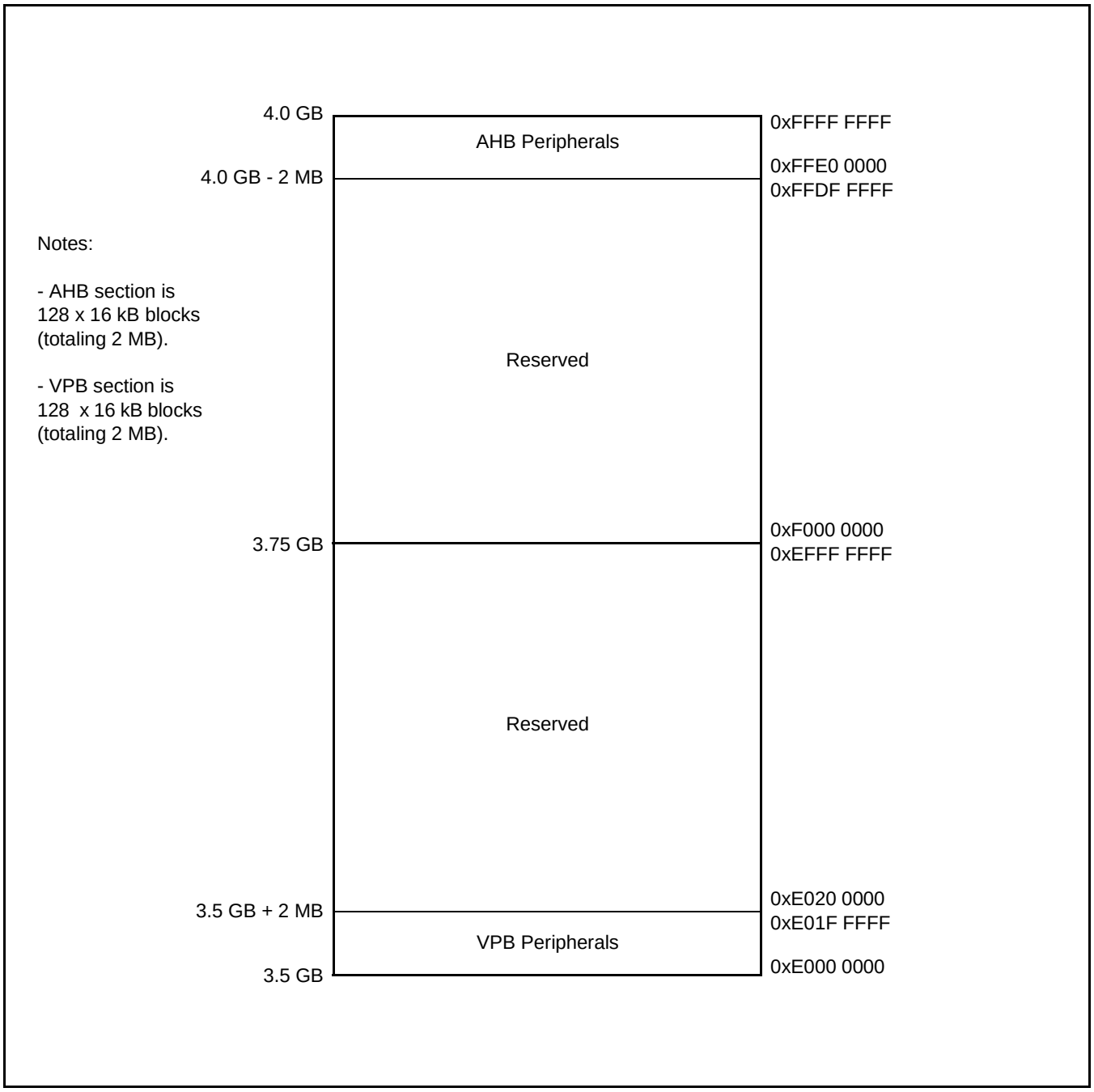


Figure 3: Peripheral Memory Map

Figures 3 through 5 show different views of the peripheral address space. Both the AHB and VPB peripheral areas are 2 megabyte spaces which are divided up into 128 peripherals. Each peripheral space is 16 kilobytes in size. This allows simplifying the address decoding for each peripheral. All peripheral register addresses are word aligned (to 32-bit boundaries) regardless of their size. This eliminates the need for byte lane mapping hardware that would be required to allow byte (8-bit) or half-word (16-bit) accesses to occur at smaller boundaries. An implication of this is that word and half-word registers must be accessed all at once. For example, it is not possible to read or write the upper byte of a word register separately.

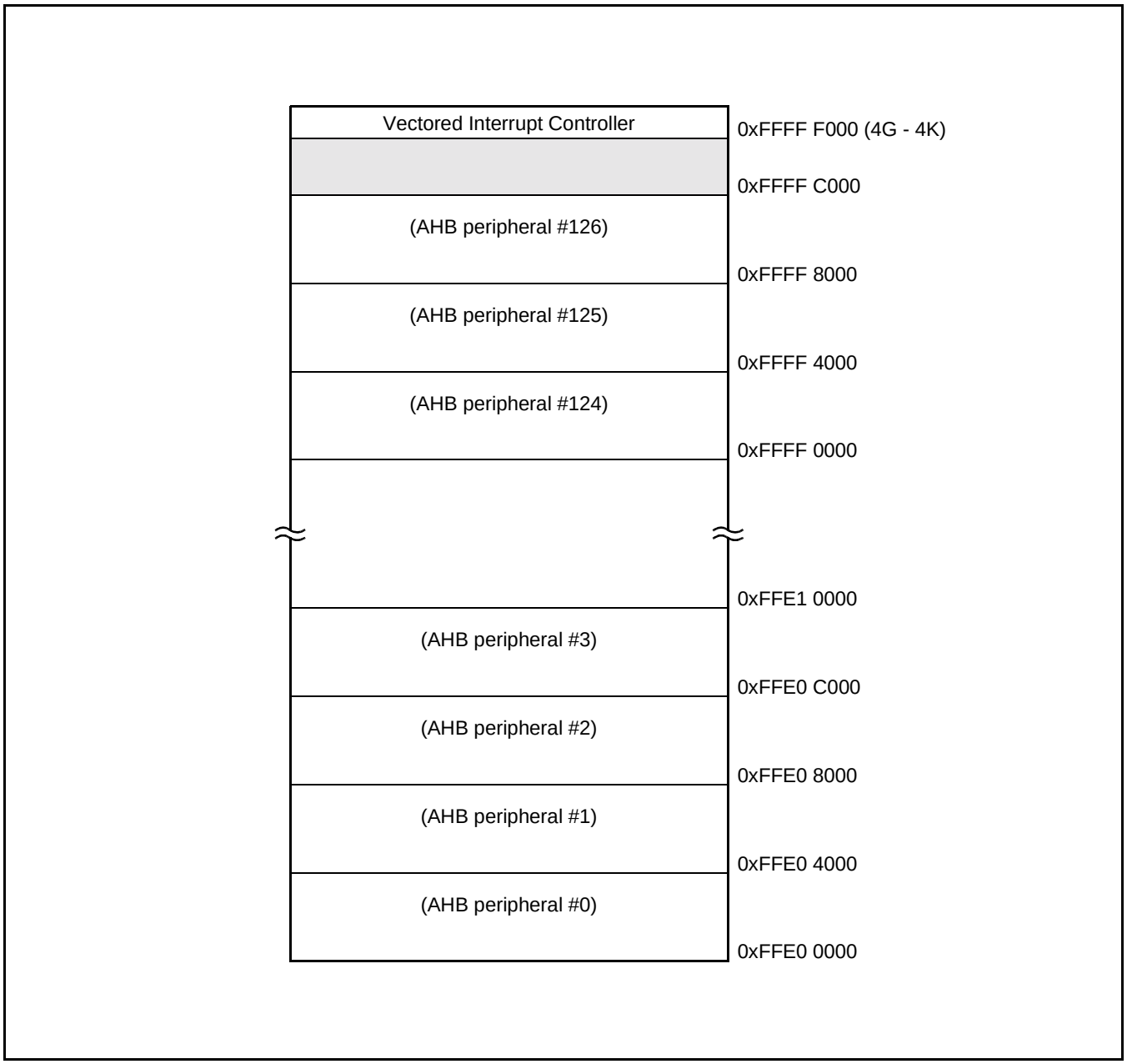


Figure 4: AHB Peripheral Map

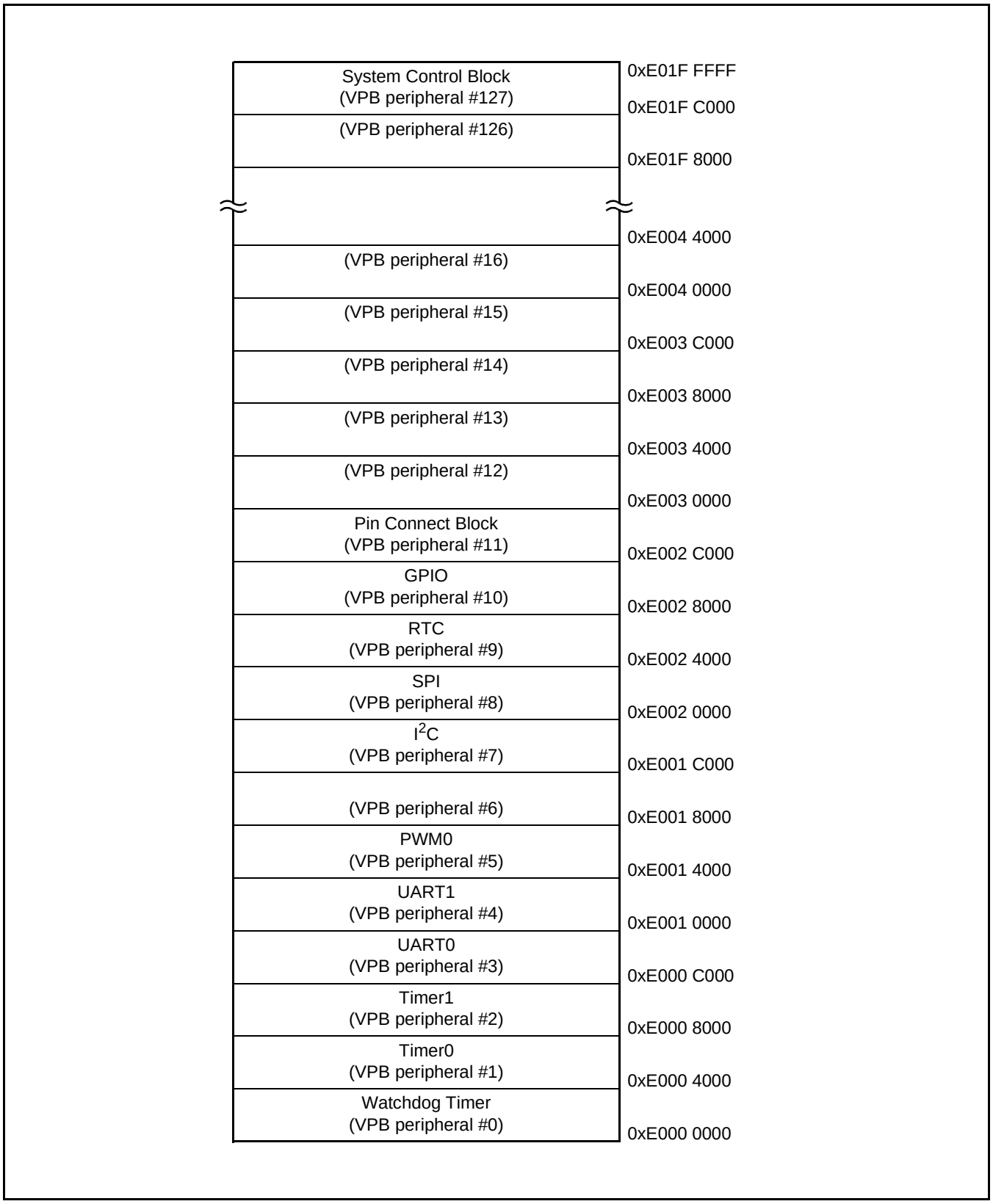


Figure 5: VPB Peripheral Map

LPC2106/2105/2104 MEMORY RE-MAPPING AND BOOT BLOCK

Memory Map Concepts and Operating Modes

The basic concept on the LPC2106/2105/2104 is that each memory area has a "natural" location in the memory map. This is the address range for which code residing in that area is written. The bulk of each memory space remains permanently fixed in the same location, eliminating the need to have portions of the code designed to run in different address ranges.

Because of the location of the interrupt vectors on the ARM7 processor (at addresses 0x0000 0000 through 0x0000 001C, as shown in Table 2 below), a small portion of the Boot Block and SRAM spaces need to be re-mapped in order to allow alternative uses of interrupts in the different operating modes described in Table 3. Re-mapping of the interrupts is accomplished via the Memory Mapping Control feature described in the System Control Block section.

Table 2: ARM Exception Vector Locations

Address	Exception
0x0000 0000	Reset
0x0000 0004	Undefined Instruction
0x0000 0008	Software Interrupt
0x0000 000C	Prefetch Abort (instruction fetch memory fault)
0x0000 0010	Data Abort (data access memory fault)
0x0000 0014	Reserved *
0x0000 0018	IRQ
0x0000 001C	FIQ

*: Identified as reserved in ARM documentation, this location is used by the Boot Loader as the Valid User Program key.

Table 3: LPC2106/2105/2104 Memory Mapping Modes

Mode	Activation	Usage
Boot Loader mode	Hardware activation by any Reset	The Boot Loader always executes after any reset. The Boot Block interrupt vectors are mapped to the bottom of memory to allow handling exceptions and using interrupts during the Boot Loading process.
User Flash mode	Software activation by Boot code	Activated by Boot Loader when a valid User Program Signature is recognized in memory and Boot Loader operation is not forced. Interrupt vectors are not re-mapped and are found in the bottom of the Flash memory.
User RAM mode	Software activation by User program	Activated by a User Program as desired. Interrupt vectors are re-mapped to the bottom of the Static RAM.

Memory Re-Mapping

In order to allow for compatibility with future derivatives, the entire Boot Block is mapped to the top of the on-chip memory space. In this manner, the use of larger or smaller flash modules will not require changing the location of the Boot Block (which would require changing the Boot Loader code itself) or changing the mapping of the Boot Block interrupt vectors. Memory spaces other than the interrupt vectors remain in fixed locations. Figure 6 shows the on-chip memory mapping in the modes defined above.

The portion of memory that is re-mapped to allow interrupt processing in different modes includes the interrupt vector area (32 bytes) and an additional 32 bytes, for a total of 64 bytes. The re-mapped code locations overlay addresses 0x0000 0000 through 0x0000 003F. A typical user program in the Flash memory can place the entire FIQ handler at address 0x0000 001C without any need to consider memory boundaries. The vector contained in the SRAM, external memory, and Boot Block must contain branches to the actual interrupt handlers, or to other instructions that accomplish the branch to the interrupt handlers.

There are three reasons this configuration was chosen:

1. To give the FIQ handler in the Flash memory the advantage of not having to take a memory boundary caused by the re-mapping into account.
2. Minimize the need to for the SRAM and Boot Block vectors to deal with arbitrary boundaries in the middle of code space.
3. To provide space to store constants for jumping beyond the range of single word branch instructions.

Re-mapped memory areas, including the Boot Block and interrupt vectors, continue to appear in their original location in addition to the re-mapped address.

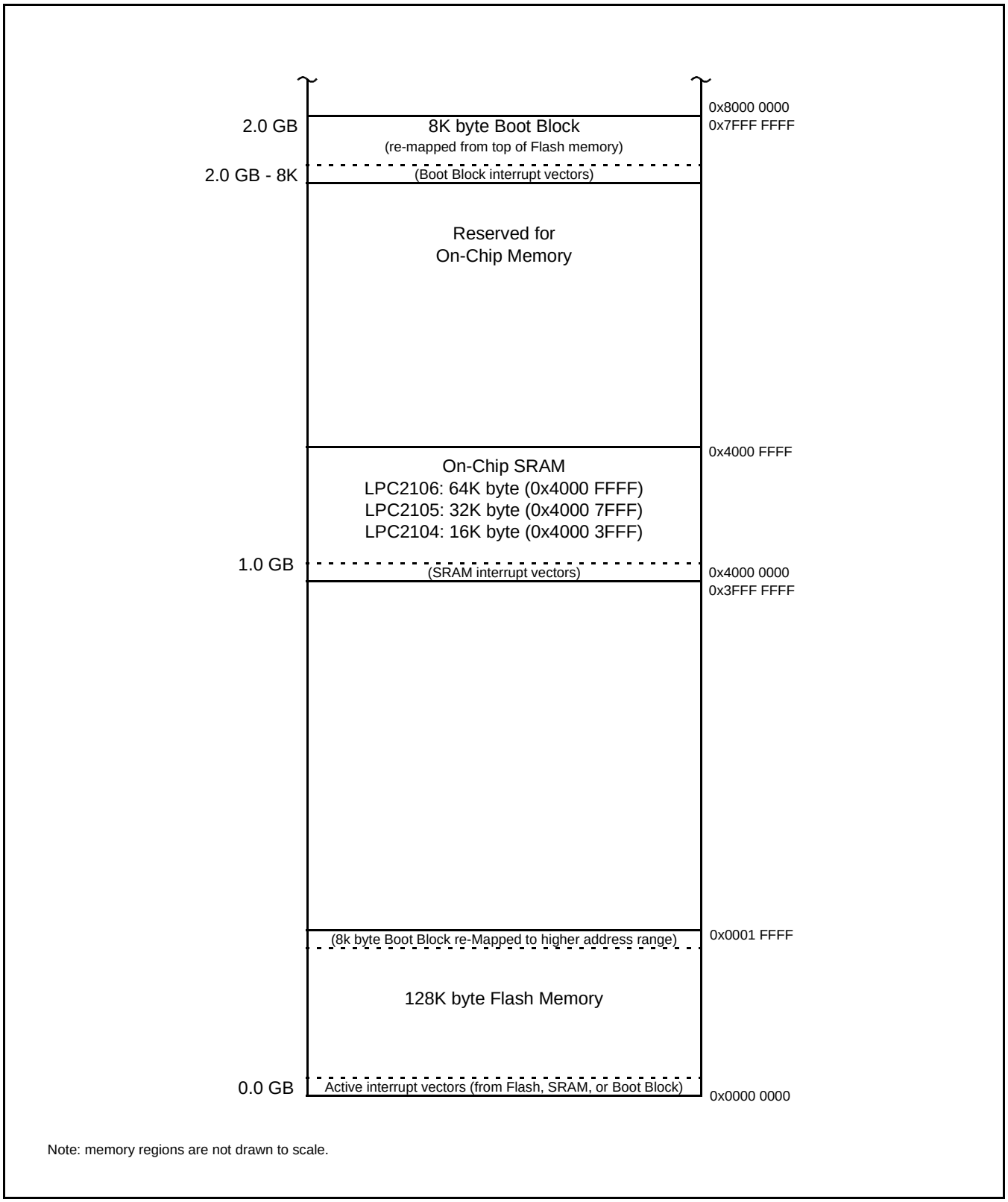


Figure 6: Map of lower memory is showing re-mapped and re-mappable areas.

PREFETCH ABORT AND DATA ABORT EXCEPTIONS

The LPC2106/2105/2104 generates the appropriate bus cycle abort exception if an access is attempted for an address that is in a reserved or unassigned address region. The regions are:

- Areas of the memory map that are not implemented for a specific ARM derivative. For the LPC2106/2105/2104, this is:
  - Address space between On-Chip Non-Volatile Memory and the Special registers. Labelled "Reserved for On-Chip Memory" in Figure 2 and Figure 6.
  - Address space between On-Chip Static RAM and External Memory. Labelled "Reserved for On-Chip Memory" in Figure 2.
  - External Memory (since no external bus interface is implemented on the LPC2106/2105/2104).
  - Reserved regions of the AHB and VPB spaces. See Figure 3.
- Unassigned AHB peripheral spaces. See Figure 4.
- Unassigned VPB peripheral spaces. See Figure 5.

For these areas, both attempted data access and instruction fetch generate an exception. In addition, a Prefetch Abort exception is generated for any instruction fetch that maps to an AHB or VPB peripheral address.

Within the address space of an existing VPB peripheral, a data abort exception is not generated in response to an access to an undefined address. Address decoding within each peripheral is limited to that needed to distinguish defined registers within the peripheral itself. For example, an access to address 0xE000D000 (an undefined address within the UART0 space) may result in an access to the register defined at address 0xE000C000. Details of such address aliasing within a peripheral space are not defined in the LPC2106/2105/2104 documentation and are not a supported feature.

Note that the ARM core stores the Prefetch Abort flag along with the associated instruction (which will be meaningless) in the pipeline and processes the abort only if an attempt is made to execute the instruction fetched from the illegal address. This prevents accidental aborts that could be caused by prefetches that occur when code is executed very near a memory boundary.

3. SYSTEM CONTROL BLOCK

SUMMARY OF SYSTEM CONTROL BLOCK FUNCTIONS

The System Control Block includes several system features and control registers for a number of functions that are not related to specific peripheral devices. These include:

- Crystal Oscillator.
- External Interrupt Inputs.
- Memory Mapping Control.
- PLL.
- Power Control.
- Reset.
- VPB Divider.
- Wakeup Timer.

Each type of function has its own register(s) if any are required and unneeded bits are defined as reserved in order to allow future expansion. Unrelated functions never share the same register addresses.

PIN DESCRIPTION

Table 4 shows pins that are associated with System Control block functions.

Table 4: Pin summary

Pin name	Pin direction	Pin Description
X1	Input	**Crystal Oscillator Input-** Input to the oscillator and internal clock generator circuits.
X2	Output	**Crystal Oscillator Output-** Output from the oscillator amplifier.
$\overline{\text{EINT0}}$	Input	**External Interrupt Input 0-** An active low general purpose interrupt input. This pin may be used to wake up the processor from Idle or Power down modes. LOW level on this pin immediately after reset is considered as an external hardware request to start the ISP command handler. More details on ISP and Flash memory can be found in "Flash Memory System and Programming" chapter.
$\overline{\text{EINT1}}$	Input	**External Interrupt Input 1-** See the EINT0 description above.
$\overline{\text{EINT2}}$	Input	**External Interrupt Input 2-** See the EINT0 description above.
$\overline{\text{RST}}$	Input	**External Reset input-** A low on this pin resets the chip, causing I/O ports and peripherals to take on their default states, and the processor to begin execution at address 0.

REGISTER DESCRIPTION

All registers, regardless of size, are on word address boundaries. Details of the registers appear in the description of each function.

Table 5: Summary of System Control Registers

Address	Name	Description	Access	Reset Value*
External Interrupts				
0xE01FC140	EXTINT	External interrupt flag register.	R/W	0
0xE01FC144	EXTWAKE	External interrupt wakeup register.	R/W	0
Memory Mapping Control				
0xE01FC040	MEMMAP	Memory mapping control.	R/W	0
Phase Locked Loop				
0xE01FC080	PLLCON	PLL control register.	R/W	0
0xE01FC084	PLLCFG	PLL configuration register.	R/W	0
0xE01FC088	PLLSTAT	PLL status register.	RO	0
0xE01FC08C	PLLFEED	PLL feed register.	WO	NA
Power Control				
0xE01FC0C0	PCON	Power control register.	R/W	0
0xE01FC0C4	PCONP	Power control for peripherals.	R/W	0x3BE
VPB Divider				
0xE01FC100	VPBDIV	VPB divider control.	R/W	0

*Reset Value refers to the data stored in used bits only. It does not include reserved bits content.

CRYSTAL OSCILLATOR

The oscillator supports crystals in the range of 10 MHz to 25 MHz. The oscillator output frequency is called F_{osc} and the ARM processor clock frequency is referred to as cclk for purposes of rate equations, etc. elsewhere in this document. F_{osc} and cclk are the same value unless the PLL is running and connected. Refer to the PLL description in this chapter for details.

Onboard oscillator in LPC2106/2105/2104 can operate in one of two modes: slave mode and oscillation mode.

In slave mode the input clock signal should be coupled by means of a capacitor of 100 pF (Cc in Figure 7, drawing a), with an amplitude of at least 200 mVrms. X2 pin in this configuration can be left not connected.

External components and models used in oscillation mode are shown in Figure 7, drawings b and c, and in Table 6. Since the feedback resistance is integrated on chip, only a crystal and the capacitances C_{X1} and C_{X2} need to be connected externally in case of fundamental mode oscillation (the fundamental frequency is represented by L, C_L and R_S). Capacitance Cp in Figure 7, drawing c, represents the parallel package capacitance and should not be larger than 7 pF. Parameters F_C, C_L, R_S and C_P are supplied by the crystal manufacturer.

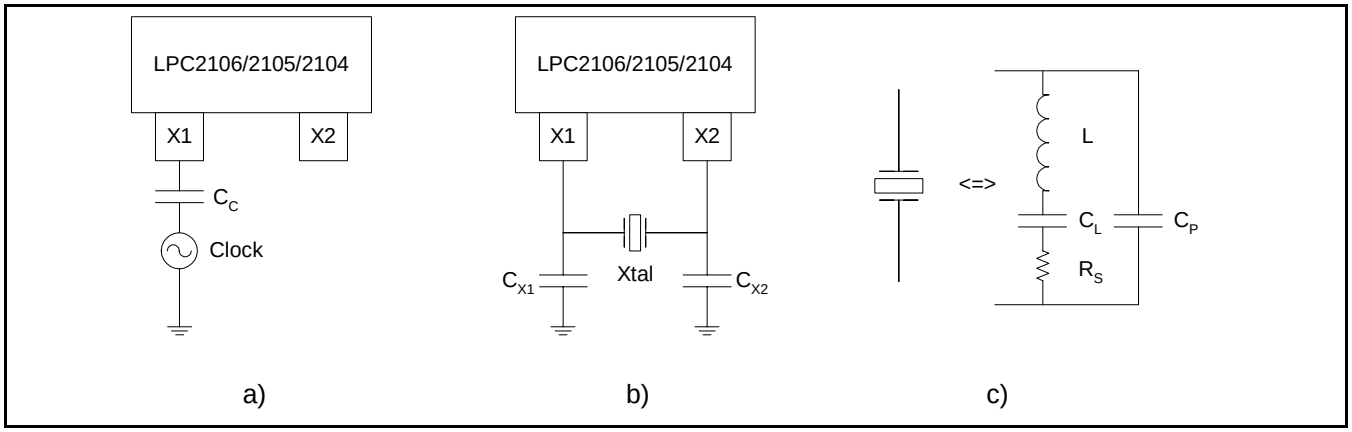


Figure 7: Oscillator modes and models: a) *slave mode* of operation, b) *oscillation mode* of operation, c) external crystal model used for $C_{X1/X2}$ evaluation

Table 6: Recommended values for $C_{X1/X2}$ when oscillation mode is used

Fundamental Oscillation Frequency F_C	Crystal Load Capacitance C_L	Max. Crystal Series Resistence R_S	External Load Capacitors C_{X1}, C_{X2}
10 - 15 MHz	10 pF	< 300 Ω	18 pF, 18 pF
	20 pF	< 220 Ω	38 pF, 38 pF
	30 pF	< 140 Ω	58 pF, 58 pF
15 - 20 MHz	10 pF	< 220 Ω	18 pF, 18 pF
	20 pF	< 140 Ω	38 pF, 38 pF
	30 pF	< 80 Ω	58 pF, 58 pF
20 - 25 MHz	10 pF	< 160 Ω	18 pF, 18 pF
	20 pF	< 90 Ω	38 pF, 38 pF
	30 pF	< 50 Ω	58 pF, 58 pF

EXTERNAL INTERRUPT INPUTS

The LPC2106/2105/2104 includes three External Interrupt Inputs as selectable pin functions. The External Interrupt Inputs can optionally be used to wake up the processor from Power Down mode.

Register Description

The external interrupt function has two registers associated with it. The EXTINT register contains the interrupt flags, and the EXTWAKEUP register contains bits that enable individual external interrupts to wake up the LPC2106/2105/2104 from Power Down mode.

Table 7: External Interrupt Registers

Address	Name	Description	Access
0xE01FC140	EXTINT	The External Interrupt Flag register contains interrupt flags for EINT0, EINT1, and EINT2. See Table 8.	R/W
0xE01FC144	EXTWAKE	The External Interrupt Wakeup register contains three enable bits that control whether each external interrupt will cause the processor to wake up from Power Down mode. See Table 9.	R/W

EXTINT Register (EXTINT - 0xE01FC140)

When an external interrupt is mapped to its related pin, the presence of a logic zero on that pin will set the corresponding interrupt flag in the EXTINT register. This will cause the VIC to respond appropriately if that interrupt is enabled. Once the logic level on external interrupt pin(s) is set to 1, software may clear the flag(s) by writing a 1 to the corresponding bit(s) in EXTINT. Every attempt to reset EINT bit is futile as long as signal level on associated pin is 0.

Table 8: External Interrupt Flag Register (EXTINT - 0xE01FC140)

EXTINT	Function	Description	Reset Value
0	EINT0	Set when external the EINT0 pin goes low and EINT0 is mapped to its related pin. Can be cleared by writing a 1 to this bit after the logic 1 appears on the related pin.	0
1	EINT1	Set when external the EINT1 pin goes low and EINT1 is mapped to its related pin. Can be cleared by writing a 1 to this bit after the logic 1 appears on the related pin.	0
2	EINT2	Set when external the EINT2 pin goes low and EINT2 is mapped to its related pin. Can be cleared by writing a 1 to this bit after the logic 1 appears on the related pin.	0
7:3	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

EXTWAKE Register (EXTWAKE - 0xE01FC144)

Enable bits in the EXTWAKE register allow the external interrupts to wake up the processor if it is in Power Down mode. The related EINTn function must be mapped to the pin in order for the wakeup process to take place. It is not necessary for the interrupt to be enabled in the Vectored Interrupt Controller for a wakeup to take place. This arrangement allows additional capabilities, such as having an external interrupt input wake up the processor from Power Down mode without causing an interrupt (simply resuming operation), or allowing an interrupt to be enabled during Power Down without waking the processor up if it is asserted (eliminating the need to disable the interrupt if the wakeup feature is not desirable in the application).

Table 9: External Interrupt Wakeup Register (EXTWAKE - 0xE01FC144)

EXTWAKE	Function	Description	Reset Value
0	EXTWAKE0	When one, assertion of $\overline{\text{EINT0}}$ will wake up the processor from Power Down mode.	0
1	EXTWAKE1	When one, assertion of $\overline{\text{EINT1}}$ will wake up the processor from Power Down mode.	0
2	EXTWAKE2	When one, assertion of $\overline{\text{EINT2}}$ will wake up the processor from Power Down mode.	0
7:3	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

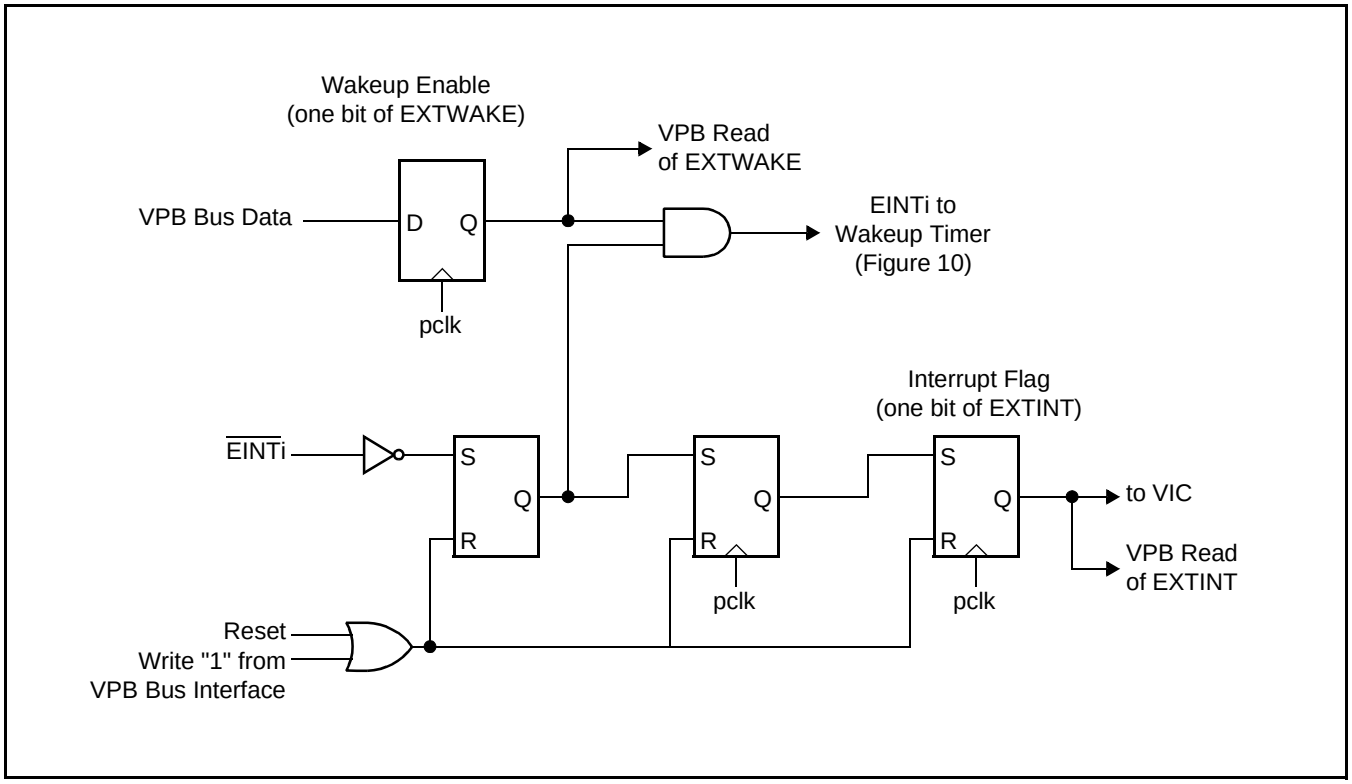


Figure 8: External Interrupt Logic

MEMORY MAPPING CONTROL

The Memory Mapping Control alters the mapping of the interrupt vectors that appear beginning at address 0x00000000. This allows code running in different memory spaces to have control of the interrupts.

Memory Mapping Control Register (MEMMAP - 0xE01FC040)

Table 10: MEMMAP Register

Address	Name	Description	Access
0xE01FC040	MEMMAP	Memory mapping control. Selects whether the ARM interrupt vectors are read from the Flash Boot Block, User Flash or RAM.	R/W

Table 11: Memory Mapping Control Register (MEMMAP - 0xE01FC040)

MEMMAP	Function	Description	Reset Value*
1:0	MAP1:0	00: Boot Loader Mode. Interrupt vectors are re-mapped to Boot Block. 01: User Flash Mode. Interrupt vectors are not re-mapped and reside in Flash. 10: User RAM Mode. Interrupt vectors are re-mapped to Static RAM. 11: Reserved. Should not be used. **Warning:** Improper setting of this value may result in incorrect operation of the device.	0
7:2	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

*: The hardware reset value of the MAP bits is 00 for LPC2106/2105/2104 parts. The apparent reset value that the user will see will be altered by the Boot Loader code, which always runs initially at reset. User documentation will reflect this difference.

PLL (PHASE LOCKED LOOP)

The PLL accepts an input clock frequency in the range of 10 MHz to 25 MHz. The input frequency is multiplied up into the range of 10 MHz to 60 MHz with a Current Controlled Oscillator (CCO). The multiplier can be an integer value from 1 to 32 (in practice, the multiplier value cannot be higher than 6 on the LPC2106/2105/2104 due to the upper frequency limit of the CPU). The CCO operates in the range of 156 MHz to 320 MHz, so there is an additional divider in the loop to keep the CCO within its frequency range while the PLL is providing the desired output frequency. The output divider may be set to divide by 2, 4, 8, or 16 to produce the output clock. Since the minimum output divider value is 2, it is insured that the PLL output has a 50% duty cycle. A block diagram of the PLL is shown in Figure 9.

PLL activation is controlled via the PLLCON register. The PLL multiplier and divider values are controlled by the PLLCFG register. These two registers are protected in order to prevent accidental alteration of PLL parameters or deactivation of the PLL. Since all chip operations, including the Watchdog Timer, are dependent on the PLL when it is providing the chip clock, accidental changes to the PLL setup could result in unexpected behavior of the microcontroller. The protection is accomplished by a feed sequence similar to that of the Watchdog Timer. Details are provided in the description of the PLLFEED register.

The PLL is turned off and bypassed following a chip Reset and when by entering power Down mode. PLL is enabled by software only. The program must configure and activate the PLL, wait for the PLL to Lock, then connect to the PLL as a clock source.

Register Description

The PLL is controlled by the registers shown in Table 12. More detailed descriptions follow.

Warning: Improper setting of PLL values may result in incorrect operation of the device.

Table 12: PLL Registers

Address	Name	Description	Access
0xE01FC080	PLLCON	Holding register for updating PLL control bits. Values written to this register do not take effect until a valid PLL feed sequence has taken place.	R/W
0xE01FC084	PLLCFG	Holding register for updating PLL configuration values. Values written to this register do not take effect until a valid PLL feed sequence has taken place.	R/W
0xE01FC088	PLLSTAT	Read-back register for PLL control and configuration information. If PLLCON or PLLCFG have been written to, but a PLL feed sequence has not yet occurred, they will not reflect the current PLL state. Reading this register provides the actual values controlling the PLL, as well as the status of the PLL.	RO
0xE01FC08C	PLLFEED	This register enables loading of the PLL control and configuration information from the PLLCON and PLLCFG registers into the shadow registers that actually affect PLL operation.	WO

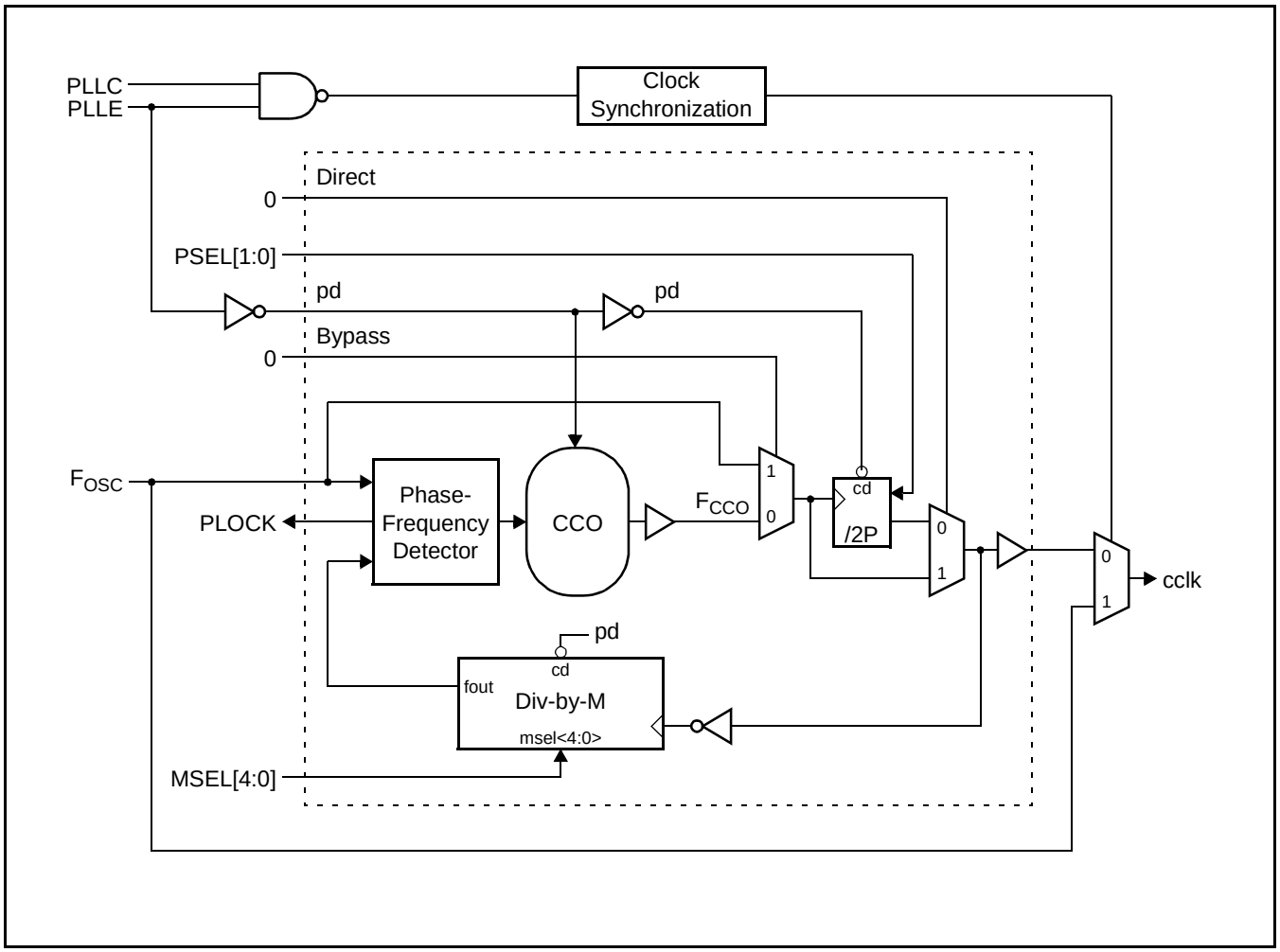


Figure 9: PLL Block Diagram

PLLCON Register (PLLCON - 0xE01FC080)

The PLLCON register contains the bits that enable and connect the PLL. Enabling the PLL allows it to attempt to lock to the current settings of the multiplier and divider values. Connecting the PLL causes the processor and all chip functions to run from the PLL output clock. Changes to the PLLCON register do not take effect until a correct PLL feed sequence has been given (see PLLFEED Register (PLLFEED - 0xE01FC08C) description).

Table 13: PLL Control Register (PLLCON - 0xE01FC080)

PLLCON	Function	Description	Reset Value
0	PLLE	PLL Enable. When one, and after a valid PLL feed, this bit will activate the PLL and allow it to lock to the requested frequency. See PLLSTAT register, Table 15.	0
1	PLLC	PLL Connect. When PLLC and PLLE are both set to one, and after a valid PLL feed, connects the PLL as the clock source for the LPC2106/2105/2104. Otherwise, the oscillator clock is used directly by the LPC2106/2105/2104. See PLLSTAT register, Table 15.	0
7:2	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

The PLL must be set up, enabled, and Lock established before it may be used as a clock source. When switching from the oscillator clock to the PLL output or vice versa, internal circuitry synchronizes the operation in order to ensure that glitches are not generated. Hardware does not insure that the PLL is locked before it is connected or automatically disconnect the PLL if lock is lost during operation. In the event of loss of PLL lock, it is likely that the oscillator clock has become unstable and disconnecting the PLL will not remedy the situation.

PLLCFG Register (PLLCFG - 0xE01FC084)

The PLLCFG register contains the PLL multiplier and divider values. Changes to the PLLCFG register do not take effect until a correct PLL feed sequence has been given (see PLLFEED Register (PLLFEED - 0xE01FC08C) description). Calculations for the PLL frequency, and multiplier and divider values are found in the PLL Frequency Calculation section.

Table 14: PLL Configuration Register (PLLCFG - 0xE01FC084)

PLLCFG	Function	Description	Reset Value
4:0	MSEL4:0	PLL Multiplier value. Supplies the value "M" in the PLL frequency calculations.	0
6:5	PSEL1:0	PLL Divider value. Supplies the value "P" in the PLL frequency calculations.	0
7	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

PLLSTAT Register (PLLSTAT - 0xE01FC088)

The read-only PLLSTAT register provides the actual PLL parameters that are in effect at the time it is read, as well as the PLL status. PLLSTAT may disagree with values found in PLLCON and PLLCFG because changes to those registers do not take effect until a proper PLL feed has occurred (see PLLFEED Register (PLLFEED - 0xE01FC08C) description).

Table 15: PLL Status Register (PLLSTAT - 0xE01FC088)

PLLSTAT	Function	Description	Reset Value
4:0	MSEL4:0	Read-back for the PLL Multiplier value. This is the value currently used by the PLL.	0
6:5	PSEL1:0	Read-back for the PLL Divider value. This is the value currently used by the PLL.	0
7	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	PLLE	Read-back for the PLL Enable bit. When one, the PLL is currently activated. When zero, the PLL is turned off. This bit is automatically cleared when Power Down mode is activated.	0
9	PLLC	Read-back for the PLL Connect bit. When PLLC and PLLE are both one, the PLL is connected as the clock source for the LPC2106/2105/2104. When zero, the PLL is bypassed and the oscillator clock is used directly by the LPC2106/2105/2104. This bit is automatically cleared when Power Down mode is activated.	0
10	PLOCK	Reflects the PLL Lock status. When zero, the PLL is not locked. When one, the PLL is locked onto the requested frequency.	0
15:11	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

PLL Interrupt

The PLOCK bit in the PLLSTAT register is connected to the interrupt controller. This allows for software to turn on the PLL and continue with other functions without having to wait for the PLL to achieve lock. When the interrupt occurs (PLOCK = 1), the PLL may be connected, and the interrupt disabled.

PLL Modes

The combinations of PLLE and PLLC are shown in Table 16.

Table 16: PLL Control Bit Combinations

PLLC	PLLE	PLL Function
0	0	PLL is turned off and disconnected. The system runs from the unmodified clock input.
0	1	The PLL is active, but not yet connected. The PLL can be connected after PLOCK is asserted.
1	0	Same as 0 0 combination. This prevents the possibility of the PLL being connected without also being enabled.
1	1	The PLL is active and has been connected as the system clock source.

PLLFEED Register (PLLFEED - 0xE01FC08C)

A correct feed sequence must be written to the PLLFEED register in order for changes to the PLLCON and PLLCFG registers to take effect. The feed sequence is:

1. Write the value 0xAA to PLLFEED
2. Write the value 0x55 to PLLFEED.

The two writes must be in the correct sequence, and must be consecutive VPB bus cycles. The latter requirement implies that interrupts must be disabled for the duration of the PLL feed operation. If either of the feed values is incorrect, or one of the previously mentioned conditions is not met, any changes to the PLLCON or PLLCFG register will not become effective.

Table 17: PLL Feed Register (PLLFEED - 0xE01FC08C)

PLLFEED	Function	Description	Reset Value
7:0	PLLFEED	The PLL feed sequence must be written to this register in order for PLL configuration and control register changes to take effect.	undefined

PLL and Power Down Mode

Power Down mode automatically turns off and disconnects the PLL. Wakeup from Power Down mode does not automatically restore the PLL settings, this must be done in software. Typically, a routine to activate the PLL, wait for lock, and then connect the PLL can be called at the beginning of any interrupt service routine that might be called due to the wakeup. It is important not to attempt to restart the PLL by simply feeding it when execution resumes after a wakeup from Power Down mode. This would enable and connect the PLL at the same time, before PLL lock is established.

PLL Frequency Calculation

The PLL equations use the following parameters:

F_{OSC} the frequency from the crystal oscillator
F_{CCO} the frequency of the PLL current controlled oscillator
cclk the PLL output frequency (also the processor clock frequency)
M PLL Multiplier value from the MSEL bits in the PLLCFG register
P PLL Divider value from the PSEL bits in the PLLCFG register

The PLL output frequency (when the PLL is both active and connected) is given by:

$$cclk = M * F_{osc} \quad \text{or} \quad cclk = \frac{F_{cco}}{2 * P}$$

The CCO frequency can be computed as:

$$F_{cco} = cclk * 2 * P \quad \text{or} \quad F_{cco} = F_{osc} * M * 2 * P$$

The PLL inputs and settings must meet the following:

- F_{osc} is in the range of 10 MHz to 25 MHz.
- cclk is in the range of 10 MHz to F_{max} (the maximum allowed frequency for the LPC2106/2105/2104).
- F_{cco} is in the range of 156 MHz to 320 MHz.

Procedure for Determining PLL Settings

If a particular application uses the PLL, its configuration may be determined as follows:

1. Choose the desired processor operating frequency (cclk). This may be based on processor throughput requirements, need to support a specific set of UART baud rates, etc. Bear in mind that peripheral devices may be running from a lower clock than the processor (see the VPB Divider description in this chapter).
2. Choose an oscillator frequency (F_{osc}). cclk must be an even multiple of F_{osc}.
3. Calculate the value of M to configure the MSEL bits. M = cclk / F_{osc}. M must be in the range of 1 to 32. The value written to the MSEL bits in PLLCFG is M - 1 (see Table 19).
4. Find a value for P to configure the PSEL bits, such that F_{cco} is within its defined frequency limits. F_{cco} is calculated using the equation given above. P must have one of the values 1, 2, 4, or 8. The value written to the PSEL bits in PLLCFG is 00 for P = 1; 01 for P = 2; 10 for P = 4; 11 for P = 8 (see Table 18).

Table 18: PLL Divider Values

PSEL Bits (PLLCFG bits 6:5)	Value of P
00	1
01	2
10	4
11	8

Table 19: PLL Multiplier Values

MSEL Bits (PLLCFG bits 4:0)	Value of M
00000	1
00001	2
00010	3
00011	4
...	...
11110	31
11111	32

POWER CONTROL

The LPC2106/2105/2104 supports two reduced power modes: Idle mode and Power Down mode. In Idle mode, execution of instructions is suspended until either a Reset or interrupt occurs. Peripheral functions continue operation during Idle mode and may generate interrupts to cause the processor to resume execution. Idle mode eliminates power used by the processor itself, memory systems and related controllers, and internal buses.

In Power Down mode, the oscillator is shut down and the chip receives no internal clocks. The processor state and registers, peripheral registers, and internal SRAM values are preserved throughout Power Down mode and the logic levels of chip pins remain static. The Power Down mode can be terminated and normal operation resumed by either a Reset or certain specific interrupts that are able to function without clocks. Since all dynamic operation of the chip is suspended, Power Down mode reduces chip power consumption to nearly zero.

Wakeup from Power Down or Idle modes via an interrupt resumes program execution in such a way that no instructions are lost, incomplete, or repeated. Wake up from Power Down mode is discussed further in the description of the Wakeup Timer later in this chapter.

A Power Control for Peripherals feature allows individual peripherals to be turned off if they are not needed in the application, resulting in additional power savings.

Register Description

The Power Control function contains two registers, as shown in Table 20. More detailed descriptions follow.

Table 20: Power Control Registers

Address	Name	Description	Access
0xE01FC0C0	PCON	Power Control Register. This register contains control bits that enable the two reduced power operating modes of the LPC2106/2105/2104. See Table 21.	R/W
0xE01FC0C4	PCONP	Power Control for Peripherals Register. This register contains control bits that enable and disable individual peripheral functions, Allowing elimination of power consumption by peripherals that are not needed. See Table 22.	R/W

PCON Register (PCON - 0xE01FC0C0)

The PCON register contains two bits. Writing a one to the corresponding bit causes entry to either the Power Down or Idle mode. If both bits are set, Power Down mode is entered.

Table 21: Power Control Register (PCON - 0xE01FC0C0)

PCON	Function	Description	Reset Value
0	IDL	Idle mode - when set, this bit causes the processor clock to be stopped, while on-chip peripherals remain active. Any enabled interrupt from a peripheral or an external interrupt source will cause the processor to resume execution.	0
1	PD	Power Down mode - when set, this bit causes the oscillator and all on-chip clocks to be stopped. A wakeup condition from an external interrupt can cause the oscillator to re-start, the PD bit to be cleared, and the processor to resume execution.	0
7:2	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Power Control for Peripherals Register (PCONP - 0xE01FC0C4)

The PCONP register allows turning off selected peripheral functions for the purpose of saving power. A few peripheral functions cannot be turned off (i.e. the Watchdog timer, GPIO, the Pin Connect block, and the System Control block). Each bit in PCONP controls one peripheral as shown in Table 22. The bit numbers correspond to the related peripheral number as shown in the VPB peripheral map in the LPC2106/2105/2104 Memory Addressing section.

Table 22: Power Control for Peripherals Register (PCONP - 0xE01FC0C4)

PCONP	Function	Description	Reset Value
0	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
1	PCTIM0	When 1, Timer 0 is enabled. When 0, Timer 0 is disabled to conserve power.	1
2	PCTIM1	When 1, Timer 1 is enabled. When 0, Timer 1 is disabled to conserve power.	1
3	PCURT0	When 1, UART 0 is enabled. When 0, UART 0 is disabled to conserve power.	1
4	PCURT1	When 1, UART 1 is enabled. When 0, UART 1 is disabled to conserve power.	1
5	PCPWM0	When 1, PWM 0 is enabled. When 0, PWM 0 is disabled to conserve power.	1
6	Reserved	User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7	PCI2C	When 1, the I^2C interface is enabled. When 0, the I^2C interface is disabled to conserve power.	1
8	PCSPI	When 1, the SPI interface is enabled. When 0, the SPI interface is disabled to conserve power.	1
9	PCRTC	When 1, the RTC is enabled. When 0, the RTC is disabled to conserve power.	1
31:10	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

RESET

Reset has two sources on the LPC2106/2105/2104: the $\overline{\text{RST}}$ pin and Watchdog Reset. The $\overline{\text{RST}}$ pin is a Schmitt trigger input pin with an additional glitch filter. Assertion of chip Reset by any source starts the Wakeup Timer (see Wakeup Timer description later in this chapter), causing reset to remain asserted until the external Reset is de-asserted, the oscillator is running, a fixed number of clocks have passed, and the Flash controller has completed its initialization. The relationship between Reset, the oscillator, and the Wakeup Timer are shown in Figure 10.

The Reset glitch filter allows the processor to ignore external reset pulses that are very short, and also determines the minimum duration of $\overline{\text{RST}}$ that must be asserted in order to guarantee a chip reset. Once asserted, $\overline{\text{RST}}$ pin can be deasserted only when crystal oscillator is fully running and an adequate signal is present on the X1 pin of the LPC2104/2105/2106. Assuming that an external crystal is used in the crystal oscillator subsystem, after power on, the $\overline{\text{RST}}$ pin should be asserted for 10 ms. For all subsequent resets when crystal osillator is already running and stable signal is on the X1 pin, the $\overline{\text{RST}}$ pin needs to be asserted for 300 ns only.

When the internal Reset is removed, the processor begins executing at address 0, which is the Reset vector. At that point, all of the processor and peripheral registers have been initialized to predetermined values.

External and internal Resets have some small differences. An external Reset causes the value of certain pins to be latched to configure the part. External circuitry cannot determine when an internal Reset occurs in order to allow setting up those special pins, so those latches are not reloaded during an internal Reset. Pins that are examined during an external Reset for various purposes are: DBGSEL, RTCK, and EINT0.

It is possible for a chip Reset to occur during a Flash programming or erase operation. The Flash memory will interrupt the ongoing operation and hold off the completion of Reset to the CPU until internal Flash high voltages have settled.

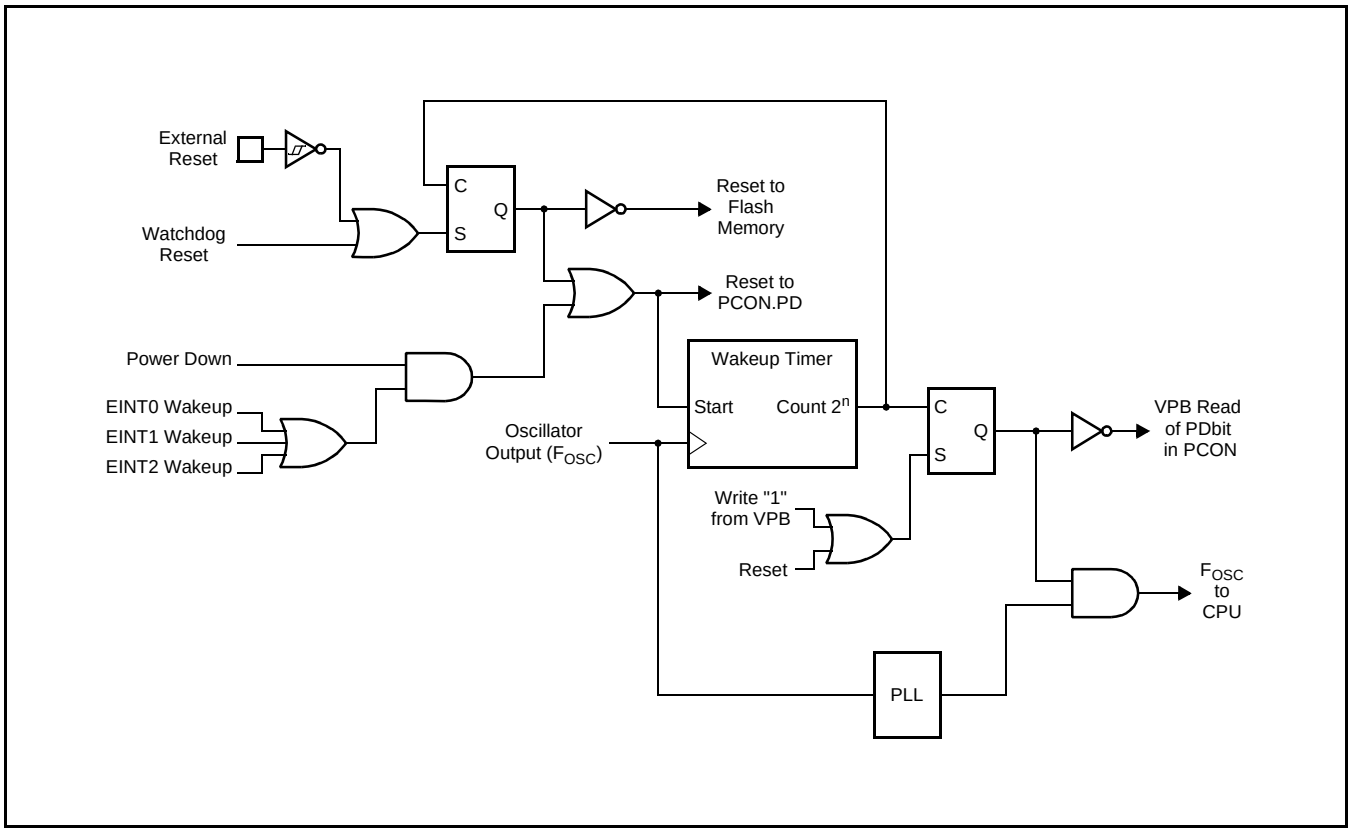


Figure 10: Reset Block Diagram including Wakeup Timer

VPB DIVIDER

The VPB Divider determines the relationship between the processor clock (cclk) and the clock used by peripheral devices (pclk). The VPB Divider serves two purposes. The first is to provides peripherals with desired pclk via VPB bus so that they can operate at the speed chosen for the ARM processor. In order to achieve this, the VPB bus may be slowed down to one half or one fourth of the processor clock rate. Because the VPB bus must work properly at power up (and its timing cannot be altered if it does not work since the VPB divider control registers reside on the VPB bus), the default condition at reset is for the VPB bus to run at one quarter speed. The second purpose of the VPB Divider is to allow power savings when an application does not require any peripherals to run at the full processor rate.

The connection of the VPB Divider relative to the oscillator and the processor clock is shown in Figure 11. Because the VPB Divider is connected to the PLL output, the PLL remains active (if it was running) during Idle mode.

VPBDIV Register (VPBDIV - 0xE01FC100)

The VPB Divider register contains two bits, allowing three divider values, as shown in Table 24.

Table 23: VPBDIV Register Map

Address	Name	Description	Access
0xE01FC100	VPBDIV	Controls the rate of the VPB clock in relation to the processor clock.	R/W

Table 24: VPB Divider Register (VPBDIV - 0xE01FC100)

VPBDIV	Function	Description	Reset Value
1:0	VPBDIV	The rate of the VPB clock is as follows: 0 0: VPB bus clock is one fourth of the processor clock. 0 1: VPB bus clock is the same as the processor clock. 1 0: VPB bus clock is one half of the processor clock. 1 1: Reserved. If this value is written to the VPBDIV register, it has no effect (the previous setting is retained).	0
7:2	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

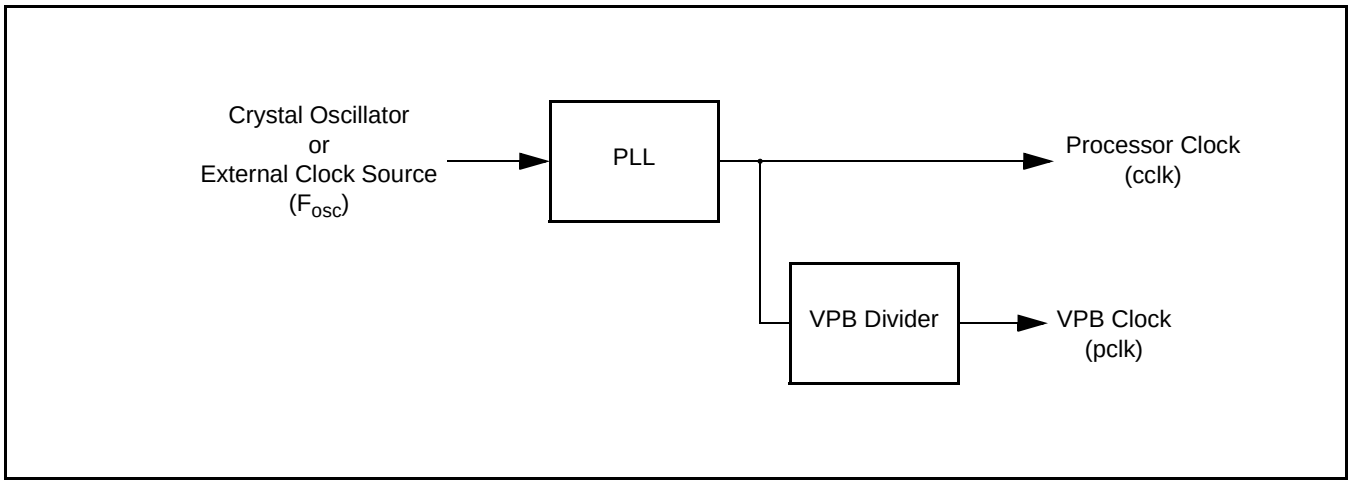


Figure 11: VPB Divider Connections

WAKEUP TIMER

The purpose of the wakeup timer is to ensure that the oscillator and other analog functions required for chip operation are fully functional before the processor is allowed to execute instructions. This is important at power on, all types of Reset, and whenever any of the aforementioned functions are turned off for any reason. Since the oscillator and other functions are turned off during Power Down mode, any wakeup of the processor from Power Down mode makes use of the Wakeup Timer.

The Wakeup Timer monitors the crystal oscillator as the means of checking whether it is safe to begin code execution. When power is applied to the chip, or some event caused the chip to exit Power down mode, some time is required for the oscillator to produce a signal of sufficient amplitude to drive the clock logic. The amount of time depends on many factors, including the rate of Vdd ramp (in the case of power on), the type of crystal and its electrical characteristics (if a quartz crystal is used), as well as any other external circuitry (e.g. capacitors), and the characteristics of the oscillator itself under the existing ambient conditions.

Once a clock is detected, the Wakeup Timer counts 4096 clocks, then enables the Flash memory to initialize. When the Flash memory initialization is complete, the processor is released to execute instructions if the external Reset has been de-asserted. In the case where an external clock source is used in the system (as opposed to a crystal connected to the oscillator pins), the possibility that there could be little or no delay for oscillator start-up must be considered. The Wakeup Timer design then ensures that any other required chip functions will be operational prior to the beginning of program execution.

The LPC2106/2105/2104 does not contain any analog function such as comparators that operate without clocks or any independent clock source such as a dedicated Watchdog oscillator. The only remaining functions that can operate in the absence of a clock source are the external interrupts, EINT0, EINT1, and EINT2. If the external interrupt is enabled to cause wakeup and becomes active (is driven low externally), an oscillator wakeup must be initiated. If the interrupt is also enabled in the Vectored Interrupt Controller, the completion of interrupt processing is postponed until the wakeup timer expires.

To summarize: on the LPC2106/2105/2104, the Wakeup Timer enforces a minimum reset duration based on the crystal oscillator, and is activated whenever there is a wakeup from Power Down mode or any type of Reset.

4. MEMORY ACCELERATOR MODULE (MAM)

INTRODUCTION

Simply put, the Memory Accelerator Module (MAM) attempts to have the next ARM instruction that will be needed in its latches in time to prevent CPU fetch stalls. The method used is to split the Flash memory into two banks, each capable of independent accesses. Each of the two Flash banks has its own prefetch Buffer and Branch Trail Buffer. The Branch Trail Buffers for the two banks capture two 128-bit lines of Flash data when an Instruction Fetch is not satisfied by either the Prefetch buffer nor Branch Trail buffer for its bank, and for which a prefetch has not been initiated. Each prefetch buffer captures one 128-bit line of instructions from its Flash bank, at the conclusion of a prefetch cycle initiated speculatively by the MAM.

Each 128 bit value includes four 32-bit ARM instructions or eight 16-bit Thumb instructions. During sequential code execution, typically one Flash bank contains or is fetching the current instruction and the entire Flash line that contains it. The other bank contains or is prefetching the next sequential code line. After a code line delivers its last instruction, the bank that contained it begins to fetch the next line in that bank.

Timing of Flash read operations is programmable and is described later in this section as well as in the System Control Block section.

Branches and other program flow changes cause a break in the sequential flow of instruction fetches described above. When a backward branch occurs, there is a distinct possibility that a loop is being executed. In this case the Branch Trail Buffers may already contain the target instruction. If so, execution continues without the need for a Flash read cycle. For a forward branch, there is also a chance that the new address is already contained in one of the Prefetch Buffers. If it is, the branch is again taken with no delay.

When a branch outside the contents of the Branch Trail and Prefetch buffers is taken, one Flash Access cycle is needed to load the Branch Trail buffers. Subsequently, there will typically be no further fetch delays until another such “Instruction Miss” occurs.

The Flash memory controller detects data accesses to the Flash memory and uses a separate buffer to store the results in a manner similar to that used during code fetches. This allows faster access to data if it is accessed sequentially. A single line buffer is provided for data accesses, as opposed to the two buffers per Flash bank that are provided for code accesses. There is no prefetch function for data accesses.

Memory Accelerator Module Blocks

The Memory Accelerator Module is divided into several functional blocks:

- A Flash Address Latch for each bank. An Incrementer function is associated with the Bank 0 Flash Address latch.
- Two Flash Memory Banks.
- Instruction Latches, Data Latches, Address Comparison latches.
- Wait logic

Figure 12 shows a simplified block diagram of the Memory Accelerator Module data paths.

In the following descriptions, the term “fetch” applies to an explicit Flash read request from the ARM. “prefetch” is used to denote a Flash read of instructions beyond the current processor fetch address.

Flash Memory Banks

There are two banks of Flash memory in order to allow two parallel accesses and eliminate delays for sequential accesses.

Flash programming operations are not controlled by the Memory Accelerator Module, but are handled as a separate function. A “boot block” sector contains Flash programming algorithms that may be called as part of the application program, and a loader that may be run to allow serial programming of the Flash memory.

The Flash memories are wired so that each sector exists in both banks, such that a sector erase operation acts on part of both banks simultaneously. In effect, the existence of two banks is transparent to the programming functions.

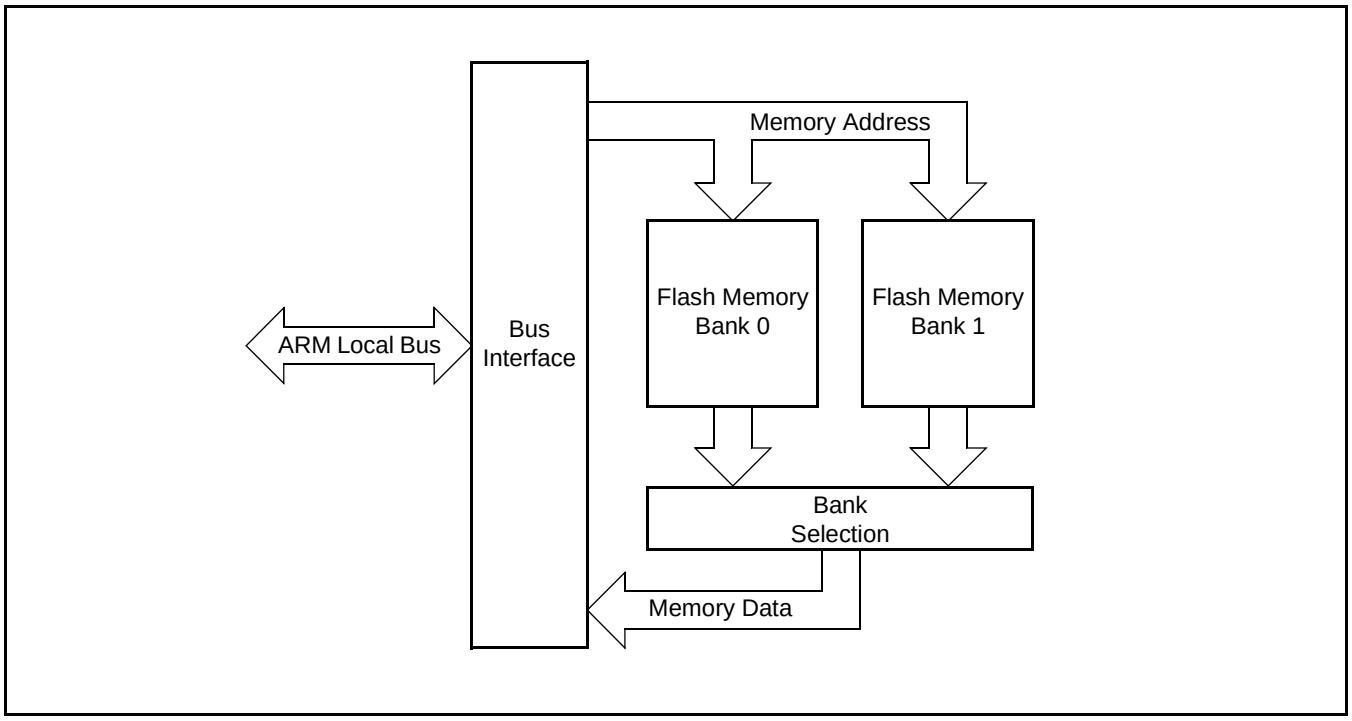


Figure 12: Simplified Block Diagram of the Memory Accelerator Module

Instruction Latches and Data Latches

Code and Data accesses are treated separately by the Memory Accelerator Module. There are two sets of 128-bit Instruction Latches and 12-bit Comparison Address Latches associated with each Flash Bank. One of the two sets, called the Branch Trail Buffer, holds the data and comparison address for that bank from the last Instruction miss. The other set, called the Prefetch Buffer, holds the data and comparison address from prefetches undertaken speculatively by the MAM. Each Instruction Latch holds 4 words of code (4 ARM instructions, or 8 Thumb instructions).

Similarly there is a 128-bit Data Latch and 13-bit Data Address latch, that are used during Data cycles. This single set of latches is shared by both Flash banks. Each Data access that is not in the Data latch causes a Flash fetch of 4 words of data, which are captured in the Data latch. This speeds up sequential Data operations, but has little or no effect on random accesses.

Flash Programming Issues

Since the Flash memory does not allow accesses during programming and erase operations, it is necessary for the MAM to force the CPU to wait if a memory access to a Flash address is requested while the Flash module is busy. (This is accomplished by asserting the ARM7TDMI-S local bus signal CLKEN.) Under some conditions, this delay could result in a Watchdog time-out. The user will need to be aware of this possibility and take steps to insure that an unwanted Watchdog reset does not cause a system failure while programming or erasing the Flash memory.

In order to preclude the possibility of stale data being read from the Flash memory, the MAM holding latches are automatically invalidated at the beginning of any Flash programming or erase operation. Any subsequent read from a Flash address will cause a new fetch to be initiated after the Flash operation has completed.

MEMORY ACCELERATOR MODULE OPERATING MODES

Three modes of operation are defined for the MAM, trading off performance for ease of predictability:

0) MAM off. All memory requests result in a Flash read operation (see note 2 below). There are no instruction prefetches.

1) MAM partially enabled. Sequential instruction accesses are fulfilled from the holding latches if the data is present. Instruction prefetch is enabled. Non-sequential instruction accesses initiate Flash read operations (see note 2 below). This means that all branches cause memory fetches. All data operations cause a Flash read because buffered data access timing is hard to predict and is very situation dependent.

2) MAM fully enabled. Any memory request (code or data) for a value that is contained in one of the corresponding holding latches is fulfilled from the latch. Instruction prefetch is enabled. Flash read operations are initiated for instruction prefetch and code or data values not available in the corresponding holding latches.

Table 25: MAM Responses to Program Accesses of Various Types

Program Memory Request Type	MAM Mode		
	0	1	2
Sequential access, data in MAM latches	Initiate Fetch [2]	Use Latched Data [1]	Use Latched Data [1]
Sequential access, data not in MAM latches	Initiate Fetch	Initiate Fetch [1]	Initiate Fetch [1]
Non-Sequential access, data in MAM latches	Initiate Fetch [2]	Initiate Fetch [1, 2]	Use Latched Data [1]
Non-Sequential access, data not in MAM latches	Initiate Fetch	Initiate Fetch [1]	Initiate Fetch [1]

Table 26: MAM Responses to Data and DMA Accesses of Various Types

Data Memory Request Type	MAM Mode		
	0	1	2
Sequential access, data in MAM latches	Initiate Fetch [2]	Initiate Fetch [2]	Use Latched Data
Sequential access, data not in MAM latches	Initiate Fetch	Initiate Fetch	Initiate Fetch
Non-Sequential access, data in MAM latches	Initiate Fetch [2]	Initiate Fetch [2]	Use Latched Data
Non-Sequential access, data not in MAM latches	Initiate Fetch	Initiate Fetch	Initiate Fetch

1. Instruction prefetch is enabled in modes 1 and 2.
2. The MAM actually uses latched data if it is available, but mimics the timing of a Flash read operation. This saves power while resulting in the same execution timing. The MAM can truly be turned off by setting the fetch timing value in MAMTIM to one clock.

MAM CONFIGURATION

After reset the MAM defaults to the disabled state. Software can turn memory access acceleration on or off at any time. This allows most of an application to be run at the highest possible performance, while certain functions can be run at a somewhat slower but more predictable rate if more precise timing is required.

REGISTER DESCRIPTION

All registers, regardless of size, are on word address boundaries. Details of the registers appear in the description of each function.

Table 27: Summary of System Control Registers

Address	Name	Description	Access	Reset Value*
MAM				
0xE01FC000	MAMCR	Memory Accelerator Module Control Register. Determines the MAM functional mode, that is, to what extent the MAM performance enhancements are enabled. See Table 28.	R/W	0
0xE01FC004	MAMTIM	Memory Accelerator Module Timing control. Determines the number of clocks used for Flash memory fetches (1 to 7 processor clocks).	R/W	0x07

*Reset Value refers to the data stored in used bits only. It does not include reserved bits content.

MAM Control Register (MAMCR - 0xE01FC000)

Two configuration bits select the three MAM operating modes, as shown in Table 28. Following Reset, MAM functions are disabled. Changing the MAM operating mode causes the MAM to invalidate all of the holding latches, resulting in new reads of Flash information as required.

Table 28: MAM Control Register (MAMCR - 0xE01FC000)

MAMCR	Function	Description	Reset Value
1:0	MAM mode control	These bits determine the operating mode of the MAM as follows: 0 0 - MAM functions disabled. 0 1 - MAM functions partially enabled. 1 0 - MAM functions fully enabled. 1 1 - reserved	0
7:2	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

MAM Timing Register (MAMTIM - 0xE01FC004)

The MAM Timing register determines how many cclk cycles are used to access the Flash memory. This allows tuning MAM timing to match the processor operating frequency. Flash access times from 1 clock to 7 clocks are possible. Single clock Flash accesses would essentially remove the MAM from timing calculations. In this case the MAM mode may be selected to optimize power usage.

Table 29: MAM Timing Register (MAMTIM - 0xE01FC004)

MAMTIM	Function	Description	Reset Value
2:0	MAM Fetch Cycle timing	These bits set the duration of MAM Flash fetch operations as follows: 0 0 0 = 0 - Reserved. 0 0 1 = 1 - MAM fetch cycles are 1 processor clock (cclk) in duration. 0 1 0 = 2 - MAM fetch cycles are 2 processor clocks (cclks) in duration. 0 1 1 = 3 - MAM fetch cycles are 3 processor clocks (cclks) in duration. 1 0 0 = 4 - MAM fetch cycles are 4 processor clocks (cclks) in duration. 1 0 1 = 5 - MAM fetch cycles are 5 processor clocks (cclks) in duration. 1 1 0 = 6 - MAM fetch cycles are 6 processor clocks (cclks) in duration. 1 1 1 = 7 - MAM fetch cycles are 7 processor clocks (cclks) in duration. **Warning:** Improper setting of this value may result in incorrect operation of the device.	0x07
7:3	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

MAM USAGE NOTES

When changing MAM timing, the MAM must first be turned off by writing a zero to MAMCR. A new value may then be written to MAMTIM. Finally, the MAM may be turned on again by writing a value (1 or 2) corresponding to the desired operating mode to MAMCR.

For system clock slower than 20 MHz, MAMTIM can be 001. For system clock between 20 MHz and 40 MHz, Flash access time is suggested to be 2 CCLKs, while in systems with system clock faster than 40 MHz, 3 CCLKs are proposed.

5. VECTORED INTERRUPT CONTROLLER (VIC)

FEATURES

- ARM PrimeCell™ Vectored Interrupt Controller
- 32 interrupt request inputs
- 16 vectored IRQ interrupts
- 16 priority levels dynamically assigned to interrupt requests
- Software interrupt generation

DESCRIPTION

The Vectored Interrupt Controller (VIC) takes 32 interrupt request inputs and programmably assigns them into 3 categories, FIQ, vectored IRQ, and non-vectored IRQ. The programmable assignment scheme means that priorities of interrupts from the various peripherals can be dynamically assigned and adjusted.

Fast Interrupt reQuest (FIQ) requests have the highest priority. If more than one request is assigned to FIQ, the VIC ORs the requests to produce the FIQ signal to the ARM processor. The fastest possible FIQ latency is achieved when only one request is classified as FIQ, because then the FIQ service routine can simply start dealing with that device. But if more than one request is assigned to the FIQ class, the FIQ service routine can read a word from the VIC that identifies which FIQ source(s) is (are) requesting an interrupt.

Vectored IRQs have the middle priority. Sixteen of the 32 requests can be assigned to this category. Any of the 32 requests can be assigned to any of the 16 vectored IRQ slots, among which slot 0 has the highest priority and slot 15 has the lowest.

Non-vectored IRQs have the lowest priority.

The VIC ORs the requests from all the vectored and non-vectored IRQs to produce the IRQ signal to the ARM processor. The IRQ service routine can start by reading a register from the VIC and jumping there. If any of the vectored IRQs are requesting, the VIC provides the address of the highest-priority requesting IRQs service routine, otherwise it provides the address of a default routine that is shared by all the non-vectored IRQs. The default routine can read another VIC register to see what IRQs are active.

All registers in the VIC are word registers. Byte and halfword reads and write are not supported.

Additional information on the Vectored Interrupt Controller is available in the ARM PrimeCell™ Vectored Interrupt Controller (PL190) documentation.

REGISTER DESCRIPTION

The VIC implements the registers shown in Table 30. More detailed descriptions follow.

Table 30: VIC Register Map

Address	Name	Description	Access	Reset Value*
0xFFFF F000	VICIRQStatus	IRQ Status Register. This register reads out the state of those interrupt requests that are enabled and classified as IRQ.	RO	0
0xFFFF F004	VICFIQStatus	FIQ Status Requests. This register reads out the state of those interrupt requests that are enabled and classified as FIQ.	RO	0
0xFFFF F008	VICRawIntr	Raw Interrupt Status Register. This register reads out the state of the 32 interrupt requests / software interrupts, regardless of enabling or classification.	RO	0
0xFFFF F00C	VICIntSelect	Interrupt Select Register. This register classifies each of the 32 interrupt requests as contributing to FIQ or IRQ.	R/W	0
0xFFFF F010	VICIntEnable	Interrupt Enable Register. This register controls which of the 32 interrupt requests and software interrupts are enabled to contribute to FIQ or IRQ.	R/W	0
0xFFFF F014	VICIntEnClr	Interrupt Enable Clear Register. This register allows software to clear one or more bits in the Interrupt Enable register.	W	0
0xFFFF F018	VICSoftInt	Software Interrupt Register. The contents of this register are ORed with the 32 interrupt requests from various peripheral functions.	R/W	0
0xFFFF F01C	VICSoftIntClear	Software Interrupt Clear Register. This register allows software to clear one or more bits in the Software Interrupt register.	W	0
0xFFFF F020	VICProtection	Protection enable register. This register allows limiting access to the VIC registers by software running in privileged mode.	R/W	0
0xFFFF F030	VICVectAddr	Vector Address Register. When an IRQ interrupt occurs, the IRQ service routine can read this register and jump to the value read.	R/W	0
0xFFFF F034	VICDefVectAddr	Default Vector Address Register. This register holds the address of the Interrupt Service routine (ISR) for non-vectored IRQs.	R/W	0
0xFFFF F100	VICVectAddr0	Vector address 0 register. Vector Address Registers 0-15 hold the addresses of the Interrupt Service routines (ISRs) for the 16 vectored IRQ slots.	R/W	0
0xFFFF F104	VICVectAddr1	Vector address 1 register	R/W	0
0xFFFF F108	VICVectAddr2	Vector address 2 register	R/W	0
0xFFFF F10C	VICVectAddr3	Vector address 3 register	R/W	0
0xFFFF F110	VICVectAddr4	Vector address 4 register	R/W	0
0xFFFF F114	VICVectAddr5	Vector address 5 register	R/W	0
0xFFFF F118	VICVectAddr6	Vector address 6 register	R/W	0
0xFFFF F11C	VICVectAddr7	Vector address 7 register	R/W	0
0xFFFF F120	VICVectAddr8	Vector address 8 register	R/W	0
0xFFFF F124	VICVectAddr9	Vector address 9 register	R/W	0

Table 30: VIC Register Map

Address	Name	Description	Access	Reset Value*
0xFFFF F128	VICVectAddr10	Vector address 10 register	R/W	0
0xFFFF F12C	VICVectAddr11	Vector address 11 register	R/W	0
0xFFFF F130	VICVectAddr12	Vector address 12 register	R/W	0
0xFFFF F134	VICVectAddr13	Vector address 13 register	R/W	0
0xFFFF F138	VICVectAddr14	Vector address 14 register	R/W	0
0xFFFF F13C	VICVectAddr15	Vector address 15 register	R/W	0
0xFFFF F200	VICVectCntl0	Vector control 0 register. Vector Control Registers 0-15 each control one of the 16 vectored IRQ slots. Slot 0 has the highest priority and slot 15 the lowest.	R/W	0
0xFFFF F204	VICVectCntl1	Vector control 1 register	R/W	0
0xFFFF F208	VICVectCntl2	Vector control 2 register	R/W	0
0xFFFF F20C	VICVectCntl3	Vector control 3 register	R/W	0
0xFFFF F210	VICVectCntl4	Vector control 4 register	R/W	0
0xFFFF F214	VICVectCntl5	Vector control 5 register	R/W	0
0xFFFF F218	VICVectCntl6	Vector control 6 register	R/W	0
0xFFFF F21C	VICVectCntl7	Vector control 7 register	R/W	0
0xFFFF F220	VICVectCntl8	Vector control 8 register	R/W	0
0xFFFF F224	VICVectCntl9	Vector control 9 register	R/W	0
0xFFFF F228	VICVectCntl10	Vector control 10 register	R/W	0
0xFFFF F22C	VICVectCntl11	Vector control 11 register	R/W	0
0xFFFF F230	VICVectCntl12	Vector control 12 register	R/W	0
0xFFFF F234	VICVectCntl13	Vector control 13 register	R/W	0
0xFFFF F238	VICVectCntl14	Vector control 14 register	R/W	0
0xFFFF F23C	VICVectCntl15	Vector control 15 register	R/W	0

*Reset Value refers to the data stored in used bits only. It does not include reserved bits content.

VIC REGISTERS

This section describes the VIC registers in the order in which they are used in the VIC logic, from those closest to the interrupt request inputs to those most abstracted for use by software. For most people, this is also the best order to read about the registers when learning the VIC.

Software Interrupt Register (VICSoftInt - 0xFFFFF018, Read/Write)

The contents of this register are ORed with the 32 interrupt requests from the various peripherals, before any other logic is applied.

Table 31: Software Interrupt Register (VICSoftInt - 0xFFFFF018, Read/Write)

VICSoftInt	Function	Reset Value
31:0	1:force the interrupt request with this bit number. 0: do not force the interrupt request with this bit number. Writing zeroes to bits in VICSoftInt has no effect, see VICSoftIntClear.	0

Software Interrupt Clear Register (VICSoftIntClear - 0xFFFFF01C, Write Only)

This register allows software to clear one or more bits in the Software Interrupt register, without having to first read it.

Table 32: Software Interrupt Clear Register (VICSoftIntClear - 0xFFFFF01C, Write Only)

VICSoftIntClear	Function	Reset Value
31:0	1: writing a 1 clears the corresponding bit in the Software Interrupt register, thus releasing the forcing of this request. 0: writing a 0 leaves the corresponding bit in VICSoftInt unchanged.	0

Raw Interrupt Status Register (VICRawIntr - 0xFFFFF008, Read Only)

This register reads out the state of the 32 interrupt requests and software interrupts, regardless of enabling or classification.

Table 33: Raw Interrupt Status Register (VICRawIntr - 0xFFFFF008, Read-Only)

VICRawIntr	Function	Reset Value
31:0	1: the interrupt request or software interrupt with this bit number is asserted. 0: the interrupt request or software interrupt with this bit number is negated.	0

Interrupt Enable Register (VICIntEnable - 0xFFFFF010, Read/Write)

This register controls which of the 32 interrupt requests and software interrupts contribute to FIQ or IRQ.

Table 34: Interrupt Enable Register (VICINtEnable - 0xFFFFF010, Read/Write)

VICIntEnable	Function	Reset Value
31:0	When this register is read, 1s indicate interrupt requests or software interrupts that are enabled to contribute to FIQ or IRQ. When this register is written, ones enable interrupt requests or software interrupts to contribute to FIQ or IRQ, zeroes have no effect. See the VICIntEnClear register (Table 46 below), for how to disable interrupts.	0

Interrupt Enable Clear Register (VICIntEnClear - 0xFFFFF014, Write Only)

This register allows software to clear one or more bits in the Interrupt Enable register, without having to first read it.

Table 35: Software Interrupt Clear Register (VICIntEnClear - 0xFFFFF014, Write Only)

VICIntEnClear	Function	Reset Value
31:0	1: writing a 1 clears the corresponding bit in the Interrupt Enable register, thus disabling interrupts for this request. 0: writing a 0 leaves the corresponding bit in VICIntEnable unchanged.	0

Interrupt Select Register (VICIntSelect - 0xFFFFF00C, Read/Write)

This register classifies each of the 32 interrupt requests as contributing to FIQ or IRQ.

Table 36: Interrupt Select Register (VICIntSelect - 0xFFFFF00C, Read/Write)

VICIntSelect	Function	Reset Value
31:0	1: the interrupt request with this bit number is assigned to the FIQ category. 0: the interrupt request with this bit number is assigned to the IRQ category.	0

IRQ Status Register (VICIRQStatus - 0xFFFFF000, Read Only)

This register reads out the state of those interrupt requests that are enabled and classified as IRQ. It does not differentiate between vectored and non-vectored IRQs.

Table 37: IRQ Status Register (VICIRQStatus - 0xFFFFF000, Read-Only)

VICIRQStatus	Function	Reset Value
31:0	1: the interrupt request with this bit number is enabled, classified as IRQ, and asserted.	0

FIQ Status Register (VICFIQStatus - 0xFFFFF004, Read Only)

This register reads out the state of those interrupt requests that are enabled and classified as FIQ. If more than one request is classified as FIQ, the FIQ service routine can read this register to see which request(s) is (are) active.

Table 38: IRQ Status Register (VICFIQStatus - 0xFFFFF004, Read-Only)

VICFIQStatus	Function	Reset Value
31:0	1: the interrupt request with this bit number is enabled, classified as FIQ, and asserted.	0

Vector Control Registers 0-15 (VICVectCntl0-15 - 0xFFFFF200-23C, Read/Write)

Each of these registers controls one of the 16 vectored IRQ slots. Slot 0 has the highest priority and slot 15 the lowest. Note that disabling a vectored IRQ slot in one of the VICVectCntl registers does not disable the interrupt itself, the interrupt is simply changed to the non-vectored form.

Table 39: Vector Control Registers (VICVectCntl0-15 - 0xFFFFF200-23C, Read/Write)

VICVectCntl0-15	Function	Reset Value
5	1: this vectored IRQ slot is enabled, and can produce a unique ISR address when its assigned interrupt request or software interrupt is enabled, classified as IRQ, and asserted.	0
4:0	The number of the interrupt request or software interrupt assigned to this vectored IRQ slot. As a matter of good programming practice, software should not assign the same interrupt number to more than one enabled vectored IRQ slot. But if this does occur, the lower-numbered slot will be used when the interrupt request or software interrupt is enabled, classified as IRQ, and asserted.	0

Vector Address Registers 0-15 (VICVectAddr0-15 - 0xFFFFF100-13C, Read/Write)

These registers hold the addresses of the Interrupt Service routines (ISRs) for the 16 vectored IRQ slots.

Table 40: Vector Address Registers (VICVectAddr0-15 - 0xFFFFF100-13C, Read/Write)

VICVectAddr0-15	Function	Reset Value
31:0	When one or more interrupt request or software interrupt is (are) enabled, classified as IRQ, asserted, and assigned to an enabled vectored IRQ slot, the value from this register for the highest-priority such slot will be provided when the IRQ service routine reads the Vector Address register (VICVectAddr).	0

Default Vector Address Register (VICDefVectAddr - 0xFFFFF034, Read/Write)

This register holds the address of the Interrupt Service routine (ISR) for non-vectored IRQs.

Table 41: Default Vector Address Register (VICDefVectAddr - 0xFFFFF034, Read/Write)

VICDefVectAddr	Function	Reset Value
31:0	When an IRQ service routine reads the Vector Address register (VICVectAddr), and no IRQ slot responds as described above, this address is returned.	0

Vector Address Register (VICVectAddr - 0xFFFFF030, Read/Write)

When an IRQ interrupt occurs, the IRQ service routine can read this register and jump to the value read.

Table 42: Vector Address Register (VICVectAddr - 0xFFFFF030, Read/Write)

VICVectAddr	Function	Reset Value
31:0	If any of the interrupt requests or software interrupts that are assigned to a vectored IRQ slot is (are) enabled, classified as IRQ, and asserted, reading from this register returns the address in the Vector Address Register for the highest-priority such slot. Otherwise it returns the address in the Default Vector Address Register. Writing to this register does not set the value for future reads from it. Rather, this register should be written near the end of an ISR, to update the priority hardware.	0

Protection Enable Register (VICProtection - 0xFFFFF020, Read/Write)

This one-bit register controls access to the VIC registers by software running in User mode.

Table 43: Protection Enable Register (VICProtection - 0xFFFFF020, Read/Write)

VICProtection	Function	Reset Value
0	1: the VIC registers can only be accessed in privileged mode. 0: VIC registers can be accessed in User or privileged mode.	0

INTERRUPT SOURCES

Table 37 lists the interrupt sources for each peripheral function. Each peripheral device has one interrupt line connected to the Vectored Interrupt Controller, but may have several internal interrupt flags. Individual interrupt flags may also represent more than one interrupt source.

Table 44: Connection of Interrupt Sources to the Vectored Interrupt Controller

Block	Flag(s)	VIC Channel #
WDT	Watchdog Interrupt (WDINT)	0
-	Reserved for software interrupts only	1
ARM Core	Embedded ICE, DbgCommRx	2
ARM Core	Embedded ICE, DbgCommTx	3
Timer 0	Match 0 - 3 (MR0, MR1, MR2, MR3) Capture 0 - 3 (CR0, CR1, CR2, CR3)	4
Timer 1	Match 0 - 3 (MR0, MR1, MR2, MR3) Capture 0 - 3 (CR0, CR1, CR2, CR3)	5
UART 0	Rx Line Status (RLS) Transmit Holding Register empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI)	6
UART 1	Rx Line Status (RLS) Transmit Holding Register empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) Modem Status Interrupt (MSI)	7
PWM0	Match 0 - 6 (MR0, MR1, MR2, MR3, MR4, MR5, MR6) Capture 0 - 3 (CR0, CR1, CR2, CR3)	8
I2C	SI (state change)	9
SPI	SPIF, MODF	10
-	reserved	11
PLL	PLL Lock (PLOCK)	12
RTC	RTCCIF (Counter Increment), RTCALF (Alarm)	13
System Control	External Interrupt 0 (EINT0)	14
System Control	External Interrupt 1 (EINT1)	15
System Control	External Interrupt 2 (EINT2)	16

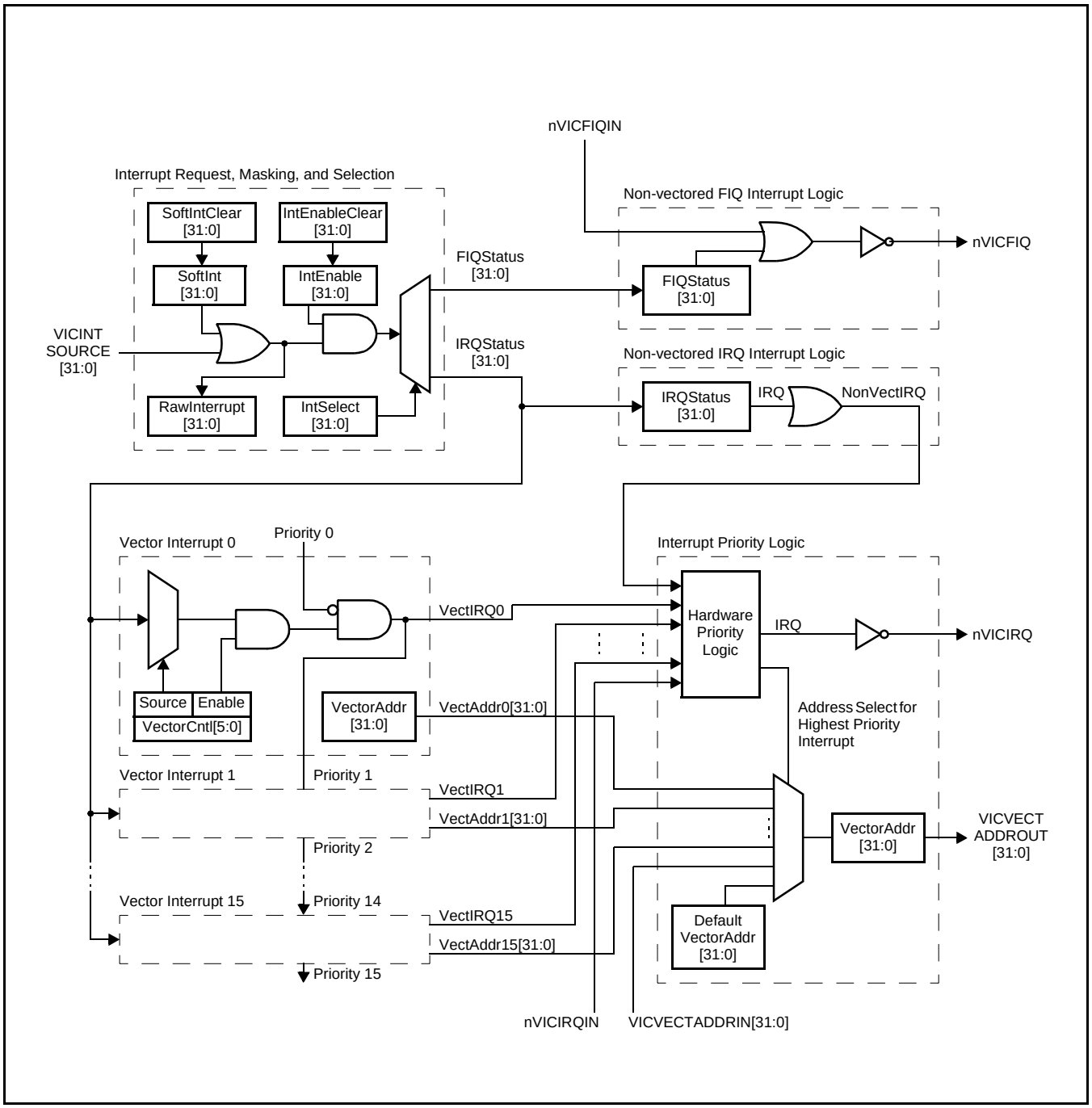


Figure 13: Block Diagram of the Vectored Interrupt Controller

VIC USAGE NOTES

If user's code is runing from the on-chip RAM and an aplication uses interrupts, interrupt vectors must be re-mapped to flash address 0x0. This is necessary because all the exception vectors are located at addresses 0x0 and above. This is easily achieved by configuring MEMMAP register (located in System Control Block) to User RAM mode. Application code should be linked such that at 0x4000 0000 the Interrupt Vector Tabe (IVT) will reside.

Although multiple sources can be selected (VICIntSelect) to generate FIQ request, only one interrupt service routine should be dedicated to service all available/present FIQ request(s). Therefore, if more than one interrupt sources are classified as FIQ the FIQ interrupt service routine must read VICFIQStatus to decide based on this content what to do and how to process the interrupt request. However, it is recommended that only one interrupt source should be classified as FIQ. Classifying more than one interrupt sources as FIQ will increase the interrupt latency.

Following the completion of the desired interrupt service routine, clearing of the interrupt flag on the peripheral level will propagate to corresponding bits in VIC registers (VICRawIntr, VICFIQStatus and VICIRQStatus). Also, before the next interrupt can be serviced, it is necessary that write is performed into the VICVectAddr register before the return from interrupt is executed. This write will clear the respective interrupt flag in the internal interrupt priority hardware.

In order to disable the interrupt at the VIC you need to clear corresponding bit in the VICIntEnClr register, which in turn clears the related bit in the VICIntEnable register. This also applies to the VICSoftInt and VICSoftIntClear in which VICSoftIntClear will clear the respective bits in VICSoftInt. For example, if VICSoftInt=0x0000 0005 and bit 0 has to be cleared, VICSoftIntClear=0x0000 0001 will acomplish this. Before the new clear operation on the same bit in VICSoftInt using writing into VICSoftIntClear is performed in the future, VICSoftIntClear=0x0000 0000 must be assigned. Therefore writing 1 to any bit in Clear register will have one-time-effect in the destination register.

If the watchdog is enabled for interrupt on underflow or invalid feed sequence only then there is no way of clearing the interrupt. The only way you could perform return from interrupt is by disabling the interrupt at the VIC(using VICIntEnClr).

Example:

Assuming that UART 0 and SPI are generating interrupt requests that are classified as vectored IRQs (UART0 being on the higher level than SPI), while UART1 and I^2C are generating non-vectored IRQs, the following could be one possibility for VIC setup:

```
VICIntSelect = 0x0000 0000(SPI, I2C, UART1 and UART0 are IRQ => bit10, bit9, bit7  and bit6=0)
VICIntEnable = 0x0000 06C0(SPI, I2C, UART1 and UART0 are enabled interrupts => bit10, bit9, bit 7 and bit6=1)
VICDefVectAddr = 0x…       (holds address at what routine for servicing non-vectored IRQs (i.e. UART1 and I2C) starts)
VICVectAddr0 = 0x…         (holds address where UART0 IRQ service routine starts)
VICVectAddr1 = 0x…         (holds address where SPI IRQ service routine starts)
VICVectCntl0 = 0x0000 0026(interrupt source with index 6 (UART0) is enabled as the one with priority 0 (the highest))
VICVectCntl1 = 0x0000 002A(interrupt source with index 10 (SPI) is enabled as the one with priority 1)
```

After any of IRQ requests (SPI, I2C, UART0 or UART1) is made, microcontroller will redirect code execution to the address specified at location 0x00000018. For vectored and non-vectored IRQ's the following instruction could be placed at 0x18:

```
LDR pc,[pc,#-0xFF0]
```

This instruction loads PC with the address that is present in VICVectAddr register.

In case UART0 request has been made, VICVectAddr will be identical to VICVectAddr0, while in case SPI request has been made value from VICVectAddr1 will be found here. If neither UART0 nor SPI have generated IRQ request but UART1 and/or I^2C were the reason, content of VICVectAddr will be identical to VICDefVectAddr.

6. PIN CONFIGURATION

LPC2106/2105/2104 PINOUT

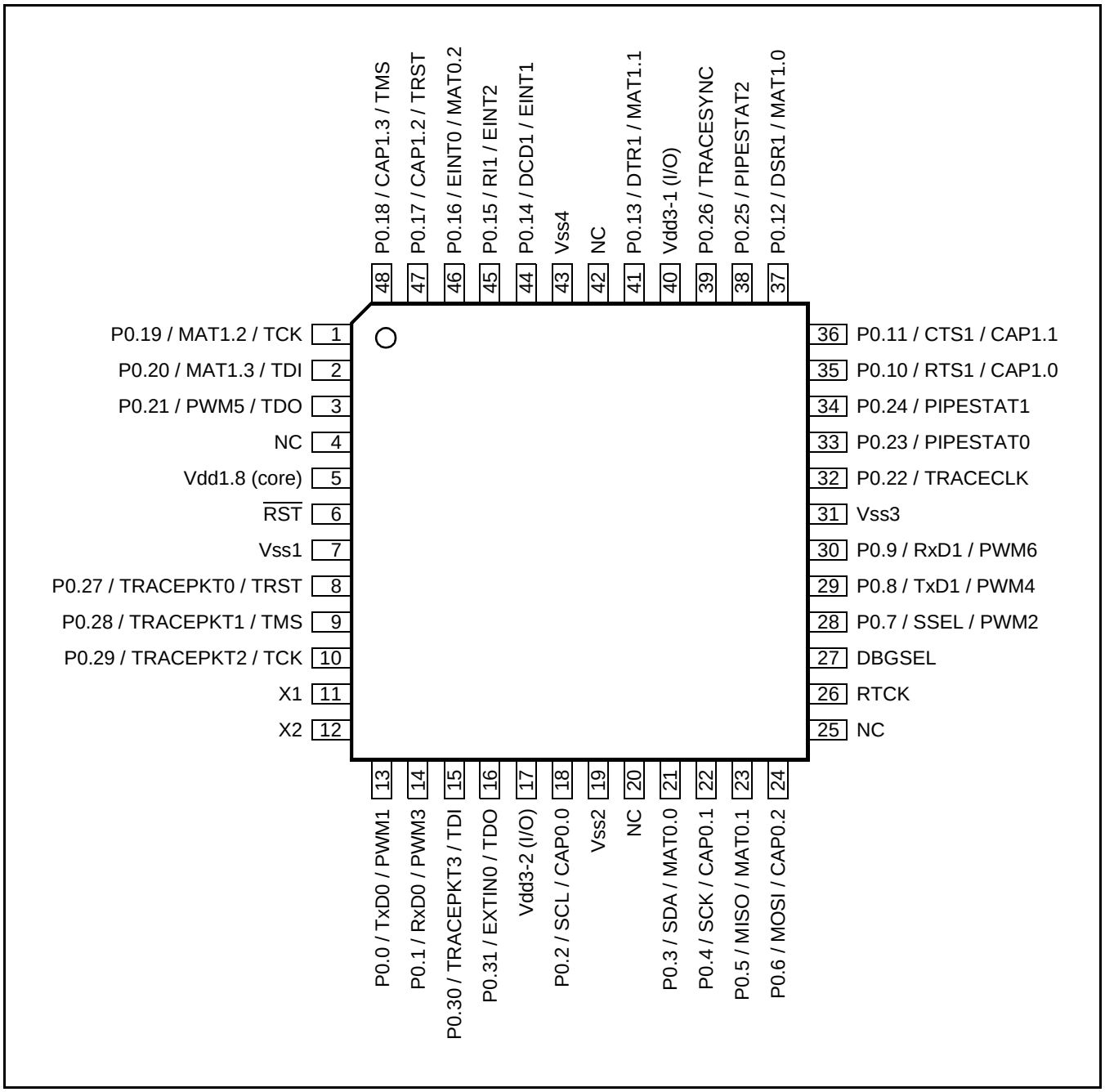


Figure 14: LPC2106/2105/2104 48-pin package (LQFP48)

LPC2106/2105/2104 PIN FUNCTIONS

Table 45: Pin description for LPC2106/2105/2104

Function Group	DBGSEL and RTCK Override[1]	Pin Name (Default Function[2])	Alternate Function 1	Alternate Function 2	Pin No.
Vdd	-	Vdd1.8	-	-	5
	-	Vdd3-1	-	-	40
Vss	-	Vss1	-	-	7
	-	Vss2	-	-	19
Oscillator	-	X1	-	-	11
	-	X2	-	-	12
Reset	-	$\overline{\text{RST}}$	-	-	6
UART0	-	P0.0	TxD0	PWM1	13
	-	P0.1	RxD0	PWM3	14
I2C	-	P0.2	SCL	CAP0.0 (Timer 0)	18
	-	P0.3	SDA	MAT0.0 (Timer 0)	21
SPI	-	P0.4	SCK	CAP0.1 (Timer 0)	22
	-	P0.5	MISO	MAT0.1 (Timer 0)	23
	-	P0.6	MOSI	CAP0.2 (Timer 0)	24
	-	P0.7	SSEL	PWM2	28
UART1	-	P0.8	TxD1	PWM4	29
	-	P0.9	RxD1	PWM6	30
	-	P0.10	RTS1	CAP1.0 (Timer 1)	35
	-	P0.11	CTS1	CAP1.1 (Timer 1)	36
	-	P0.12	DSR1	MAT1.0 (Timer 1)	37
	-	P0.13	DTR1	MAT1.1 (Timer 1)	41
	-	P0.14	DCD1	EINT1	44
	-	P0.15	RI1	EINT2	45
EXT INT	-	P0.16	EINT0	MAT0.2 (Timer 0)	46
JTAG	TRST	P0.17	CAP1.2 (Timer 1)	-	47
	TMS	P0.18	CAP1.3 (Timer 1)	-	48
	TCK	P0.19	MAT1.2 (Timer 1)	-	1
	TDI	P0.20	MAT1.3 (Timer 1)	-	2
	TDO	P0.21	PWM5	-	3
DEBUG	-	DBGSEL[1]	-	-	27
	-	Vss3	-	-	31
JTAG	-	RTCK[3]	-	-	26
ETM	TRACECLK	P0.22	-	-	32
	PIPESTAT0	P0.23	-	-	33
	PIPESTAT1	P0.24	-	-	34
	PIPESTAT2	P0.25	-	-	38
	TRACESYNC	P0.26	-	-	39
	TRACEPKT0	P0.27	TRST	-	8
	TRACEPKT1	P0.28	TMS	-	9
	TRACEPKT2	P0.29	TCK	-	10
	TRACEPKT3	P0.30	TDI	-	15
	EXTIN0	P0.31	TDO	-	16
Vss	-	Vss4	-	-	43
Vdd	-	Vdd3-2	-	-	17

1. There must be a low level at the DBGSEL input for normal operation. The DBGSEL pin has a built-in pulldown that will provide this if the pin is left unconnected in the application. Details of Debug mode may be found in "EmbeddedICE Logic" chapter.
2. This column shows the default functionality of each pin during debug mode with the primary JTAG port.
3. RTCK is an extra signal added to the JTAG port. Multi-ICE (Development system from ARM) uses this signal to maintain synchronization with targets having slow or widely varying clock frequency. For details refer to "Multi-ICE System Design considerations Application Note 72 (ARM DAI 0072A)". RTCK is used as an input when enabling debug mode to choose debug port options. Details of Debug mode may be found later in "EmbeddedICE Logic" chapter.

PIN DESCRIPTION FOR LPC2106/2105/2104

Pin description for LPC2106/2105/2104 and a brief of corresponding functions are shown in the following table.

Table 46: Pin description and corresponding functions for LPC2106/2105/2104

Pin Name	LQFP 48 Pin #	Type	Description		
P0.0 to P0.31		I/O	**Port 0:** Port 0 is a 32-bit bi-directional I/O port with individual direction controls for each bit. The operation of port 0 pins depends upon the pin function selected via the Pin Connect Block.		
	13	O	**P0.0**	**TxD0**	Transmitter output for UART 0.
		O		**PWM1**	Pulse Width Modulator output 1.
	14	I	**P0.1**	**RxD0**	Receiver input for UART 0.
		O		**PWM3**	Pulse Width Modulator output 3.
	18	I/O	**P0.2**	**SCL**	I^2C clock input/output. Open drain output (for I^2C compliance).
		I		**CAP0.0**	Capture input for Timer 0, channel 0.
	21	I/O	**P0.3**	**SDA**	I^2C data input/output. Open drain output (for I^2C compliance).
		O		**MAT0.0**	Match output for Timer 0, channel 0.
	22	I/O	**P0.4**	**SCK**	Serial Clock. SPI clock output from master or input to slave.
		I		**CAP0.1**	Capture input for Timer 0, channel 1.
	23	I/O	**P0.5**	**MISO**	Master In Slave Out. Data input to SPI master or data output from SPI slave.
		O		**MAT0.1**	Match output for Timer 0, channel 1.
	24	I/O	**P0.6**	**MOSI**	Master Out Slave In. Data output from SPI master or data input to SPI slave.
		I		**CAP0.2**	Capture input for Timer 0, channel 2.
	28	I	**P0.7**	**SSEL**	Slave Select. Selects the SPI interface as a slave.
		O		**PWM2**	Pulse Width Modulator output 2.
	29	O	**P0.8**	**TxD1**	Transmitter output for UART 1.
		O		**PWM4**	Pulse Width Modulator output 4.
	30	I	**P0.9**	**RxD1**	Receiver input for UART 1.
		O		**PWM6**	Pulse Width Modulator output 6.

Table 46: Pin description and corresponding functions for LPC2106/2105/2104

Pin Name	LQFP 48 Pin #	Type	Description		
	35	O	**P0.10**	**RTS1**	Request to Send output for UART 1.
		I		**CAP1.0**	Capture input for Timer 1, channel 0.
	36	I	**P0.11**	**CTS1**	Clear to Send input for UART 1.
		I		**CAP1.1**	Capture input for Timer 1, channel 1.
	37	I	**P0.12**	**DSR1**	Data Set Ready input for UART 1.
		O		**MAT1.0**	Match output for Timer 1, channel 0.
	41	O	**P0.13**	**DTR1**	Data Terminal Ready output for UART 1.
		O		**MAT1.1**	Match output for Timer 1, channel 1.
	44	I	**P0.14**	**DCD1**	Data Carrier Detect input for UART 1.
		I		**EINT1**	External interrupt 1 input.
	45	I	**P0.15**	**RI1**	Ring Indicator input for UART 1.
		I		**EINT2**	External interrupt 2 input.
	46	I	**P0.16**	**EINT0**	External interrupt 0 input.
		O		**MAT0.2**	Match output for Timer 0, channel 2.
	47	I	**P0.17**	**CAP1.2**	Capture input for Timer 1, channel 2.
		I		**TRST**	Test Reset for JTAG interface, primary JTAG pin group.
	48	I	**P0.18**	**CAP1.3**	Capture input for Timer 1, channel 3.
		I		**TMS**	Test Mode Select for JTAG interface, primary JTAG pin group.
	1	O	**P0.19**	**MAT1.2**	Match output for Timer 1, channel 2.
		I		**TCK**	Test Clock for JTAG interface, primary JTAG pin group.
	2	O	**P0.20**	**MAT1.3**	Match output for Timer 1, channel 3.
		I		**TDI**	Test Data In for JTAG interface, primary JTAG pin group.
	3	O	**P0.21**	**PWM5**	Pulse Width Modulator output 5.
		O		**TDO**	Test Data Out for JTAG interface, primary JTAG pin group.
	32	O	**P0.22**	**TRACECLK**	Trace Clock. Standard I/O port with internal pullup.
	33	O	**P0.23**	**PIPESTAT0**	Pipeline Status, bit 0. Standard I/O port with internal pullup.
	34	O	**P0.24**	**PIPESTAT1**	Pipeline Status, bit 1. Standard I/O port with internal pullup.
	38	O	**P0.25**	**PIPESTAT2**	Pipeline Status, bit 2. Standard I/O port with internal pullup.
	39	O	**P0.26**	**TRACESYNC**	Trace Synchronization Standard I/O port with internal pullup.

Table 46: Pin description and corresponding functions for LPC2106/2105/2104

Pin Name	LQFP48 Pin #	Type	Description
	8	O I	**P0.27** **TRACEPKT0** Trace Packet, bit 0. Standard I/O port with internal pullup. **TRST** Test Reset for JTAG interface, secondary JTAG pin group.
	9	O I	**P0.28** **TRACEPKT1** Trace Packet, bit 1. Standard I/O port with internal pullup. **TMS** Test Mode Select for JTAG interface, secondary JTAG pin group.
	10	O I	**P0.29** **TRACEPKT2** Trace Packet, bit 2. Standard I/O port with internal pullup. **TCK** Test Clock for JTAG interface, secondary JTAG pin group.
	15	O I	**P0.30** **TRACEPKT3** Trace Packet, bit 3. Standard I/O port with internal pullup. **TDI** Test Data In for JTAG interface, secondary JTAG pin group.
	16	I O	**P0.31** **EXTIN0** External Trigger Input. Standard I/O port with internal pullup. **TDO** Test Data Out for JTAG interface, secondary JTAG pin group.
RTCK	26	I/O	Returned Test Clock output. Extra signal added to the JTAG port. Assists debugger synchronization when processor frequency varies. Also used during debug mode entry to enable primary JTAG pins. Bi-directional pin with internal pullup.
DBGSEL	27	I	Debug Select. When low, the part operates normally. When high, debug mode is entered. Input pin with internal pulldown.
$\overline{RST}$	6	I	External Reset input. A low on this pin resets the device, causing I/O ports and peripherals to take on their default states, and processor execution to begin at address 0.
X1	11	I	Input to the oscillator circuit and internal clock generator circuits.
X2	12	O	Output from the oscillator amplifier.
V_{SS}	7, 19, 31, 43	I	**Ground:** 0V reference.
$V_{DD1.8}$	5	I	**1.8V Core Power Supply:** This is the power supply voltage for internal circuitry.
V_{DD3}	17, 40	I	**3.3V Pad Power Supply:** This is the power supply voltage for the I/O ports.
NC	4, 20, 25, 42	-	**Not Connected:** These pins are not connected.

7. PIN CONNECT BLOCK

FEATURES

- Allows individual pin configuration

APPLICATIONS

The purpose of the Pin Connect Block is to configure the microcontroller pins to the desired functions.

DESCRIPTION

The pin connect block allows selected pins of the microcontroller to have more than one function. Configuration registers control the multiplexers to allow connection between the pin and the on chip peripherals.

Peripherals should be connected to the appropriate pins prior to being activated, and prior to any related interrupt(s) being enabled. Activity of any enabled peripheral function that is not mapped to a related pin should be considered undefined.

REGISTER DESCRIPTION

The Pin Control Module contains 2 registers as shown in Table 47. below.

Table 47: Pin Connect Block Register Map

Address	Name	Description	Access
0xE002C000	PINSEL0	Pin function select register 0	Read/Write
0xE002C004	PINSEL1	Pin function select register 1	Read/Write

Pin Function Select Register 0 (PINSEL0 - 0xE002C000)

The PINSEL0 register controls the functions of the pins as per the settings listed in Table 50. The direction control bit in the IODIR register is effective only when the GPIO function is selected for a pin. For other functions, direction is controlled automatically.

Table 48: Pin Function Select Register 0 (PINSEL0 - 0xE002C000)

PINSEL0	Pin Name	Function when 00	Function when 01	Function when 10	Function when 11	Reset Value
1:0	P0.0	GPIO Port 0.0	TxD (UART 0)	PWM1	Reserved	0
3:2	P0.1	GPIO Port 0.1	RxD (UART 0)	PWM3	Reserved	0
5:4	P0.2	GPIO Port 0.2	SCL (I^2C)	Capture 0.0 (Timer 0)	Reserved	0
7:6	P0.3	GPIO Port 0.3	SDA (I^2C)	Match 0.0 (Timer 0)	Reserved	0
9:8	P0.4	GPIO Port 0.4	SCK (SPI)	Capture 0.1 (Timer 0)	Reserved	0
11:10	P0.5	GPIO Port 0.5	MISO (SPI)	Match 0.1 (Timer 0)	Reserved	0
13:12	P0.6	GPIO Port 0.6	MOSI (SPI)	Capture 0.2 (Timer 0)	Reserved	0
15:14	P0.7	GPIO Port 0.7	SSEL (SPI)	PWM2	Reserved	0
17:16	P0.8	GPIO Port 0.8	TxD UART 1	PWM4	Reserved	0
19:18	P0.9	GPIO Port 0.9	RxD (UART 1)	PWM6	Reserved	0
21:20	P0.10	GPIO Port 0.10	RTS (UART1)	Capture 1.0 (Timer 1)	Reserved	0
23:22	P0.11	GPIO Port 0.11	CTS (UART1)	Capture 1.1 (Timer 1)	Reserved	0
25:24	P0.12	GPIO Port 0.12	DSR (UART1)	Match 1.0 (Timer 1)	Reserved	0
27:26	P0.13	GPIO Port 0.13	DTR (UART 1)	Match 1.1 (Timer 1)	Reserved	0
29:28	P0.14	GPIO Port 0.14	CD (UART 1)	EINT1	Reserved	0
31:30	P0.15	GPIO Port 0.15	RI (UART1)	EINT2	Reserved	0

Pin Function Select Register 1 (PINSEL1 - 0xE002C004)

The PINSEL1 register controls the functions of the pins as per the settings listed in Table 49. The direction control bit in the IODIR register is effective only when the GPIO function is selected for a pin. For other functions direction is controlled automatically. Function control for the pins P0.17 - P0.31 is effective only when the DBGSEL input is pulled LOW during RESET.

Table 49: Pin Function Select Register 1 (PINSEL1 - 0xE002C004)

PINSEL1	Pin Name	Function when 00	Function when 01	Function when 10	Function when 11	Reset Value
1:0	P0.16	GPIO Port 0.16	EINT0	Match 0.2 (Timer 0)	Reserved	0
3:2	P0.17	GPIO Port 0.17	Capture 1.2 (Timer 1)	Reserved	Reserved	0
5:4	P0.18	GPIO Port 0.18	Capture 1.3 (Timer 1)	Reserved	Reserved	0
7:6	P0.19	GPIO Port 0.19	Match 1.2 (Timer 1)	Reserved	Reserved	0
9:8	P0.20	GPIO Port 0.20	Match 1.3 (Timer 1)	Reserved	Reserved	0
11:10	P0.21	GPIO Port 0.21	PWM5	Reserved	Reserved	0
13:12	P0.22	GPIO Port 0.22	Reserved	Reserved	Reserved	0
15:14	P0.23	GPIO Port 0.23	Reserved	Reserved	Reserved	0
17:16	P0.24	GPIO Port 0.24	Reserved	Reserved	Reserved	0
19:18	P0.25	GPIO Port 0.25	Reserved	Reserved	Reserved	0
21:20	P0.26	GPIO Port 0.26	Reserved	Reserved	Reserved	0
23:22	P0.27	GPIO Port 0.27	TRST	Reserved	Reserved	0
25:24	P0.28	GPIO Port 0.28	TMS	Reserved	Reserved	0
27:26	P0.29	GPIO Port 0.29	TCK	Reserved	Reserved	0
29:28	P0.30	GPIO Port 0.30	TDI	Reserved	Reserved	0
31:30	P0.31	GPIO Port 0.31	TDO	Reserved	Reserved	0

Pin Function Select Register Values

The PINSEL registers control the functions of device pins as shown below. Pairs of bits in these registers correspond to specific device pins.

Table 50: Pin Function Select Register Bits

Pinsel0 and Pinsel1 Values		Function	Value after Reset
0	0	Primary (default) function, typically GPIO Port	00
0	1	First alternate function	
1	0	Second alternate function	
1	1	Reserved	

The direction control bit in the IODIR register is effective only when the GPIO function is selected for a pin. For other functions, direction is controlled automatically. Each derivative typically has a different pinout and therefore a different set of functions possible for each pin. Details for a specific derivative may be found in the appropriate data sheet.

8. GPIO

FEATURES

- Direction control of individual bits
- Separate control of output set and clear
- All I/O default to inputs after reset

APPLICATIONS

- General purpose I/O
- Driving LEDs, or other indicators
- Controlling off-chip devices
- Sensing digital inputs

PIN DESCRIPTION

Table 51: GPIO Pin Description

Pin Name	Type	Description
P0.0 - P0.31	Input/ Output	General purpose input/output. The number of GPIOs actually available depends on the use of alternate functions.

REGISTER DESCRIPTION

The GPIO contains 4 registers as shown in Table 52.

Table 52: GPIO Register Map

Address	Name	Description	Access
0xE0028000	IOPIN	GPIO Pin value register. The current state of the port pins can always be read from this register, regardless of pin direction and mode.	Read Only
0xE0028004	IOSET	GPIO 0 Output set register. This register controls the state of output pins in conjunction with the IOCLR register. Writing ones produces highs at the corresponding port pins. Writing zeroes has no effect.	Read/Set
0xE0028008	IODIR	GPIO 0 Direction control register. This register individually controls the direction of each port pin.	Read/Write
0xE002800C	IOCLR	GPIO 0 Output clear register. This register controls the state of output pins. Writing ones produces lows at the corresponding port pins and clears the corresponding bits in the IOSET register. Writing zeroes has no effect.	Clear Only

GPIO Pin Value Register (IOPIN - 0xE0028000)

This register provides the value of the GPIO pins. This value reflects any outside world influence on the pins.

Note: for test purposes, writing to this register stores the value in the output register, bypassing the need to use both the IOSET and IOCLR registers. This feature is of little or no use in an application because it is not possible to write to individual bytes in this register.

Table 53: GPIO Pin Value Register (IOPIN - 0xE0028000)

IOPIN	Description	Value after Reset
31:0	GPIO pin value bits. Bit 0 corresponds to P0.0 ... Bit 31 corresponds to P0.31	Undefined

GPIO Output Set Register (IOSET - 0xE0028004)

This register is used to produce a HIGH level output at the port pins if they are configured as GPIO in an OUTPUT mode. Writing 1 produces a HIGH level at the corresponding port pins. Writing 0 has no effect. If any pin is configured as an input or a secondary function, writing to IOSET has no effect.

Reading the IOSET register returns the value in the GPIO output register, as determined by previous writes to IOSET and IOCLR (or IOPIN as noted above). This value does not reflect the effect of any outside world influence on the I/O pins.

Table 54: GPIO Output Set Register (IOSET - 0xE0028004)

IOSET	Description	Value after Reset
31:0	Output value SET bits. Bit 0 corresponds to P0.0 ... Bit 31 corresponds to P0.31	0

GPIO Output Clear Register (IOCLR - 0xE002800C)

This register is used to produce a LOW level at port pins if they are configured as GPIO in an OUTPUT mode. Writing 1 produces a LOW level at the corresponding port pins and clears the corresponding bits in the IOSET register. Writing 0 has no effect. If any pin is configured as an input or a secondary function, writing to IOCLR has no effect.

Table 55: GPIO Output Clear Register (IOCLR - 0xE002800C)

IOCLR	Description	Value after Reset
31:0	Output value CLEAR bits. Bit 0 corresponds to P0.0 ... Bit 31 corresponds to P0.31	0

GPIO Direction Register (IODIR - 0xE0028008)

This register is used to control the direction of the pins when they are configured as GPIO port pins. Direction bit for any pin must be set according to the pin functionality.

Table 56: GPIO Direction Register (IODIR - 0xE0028008)

IODIR	Description	Value after Reset
31:0	Direction control bits (0 = INPUT, 1 = OUTPUT). Bit 0 controls P0.0 ... Bit 31 controls P0.31	0

GPIO USAGE NOTES

If for the specified output pin corresponding bit is set both in GPIO Output Set Register (IOSET) and in GPIO Output Clear Register (IOCLR), observed pin will output level determined by the later write access of IOSET nad IOCLR. This means that in case of sequence:

IOSET = 0x0000 0080

IOCLR = 0x0000 0080

pin P0.7 will have low output, since access to Clear register came after access to Set register.

9. UART 0

FEATURES

- 16 byte Receive and Transmit FIFOs.
- Register locations conform to ‘550 industry standard.
- Receiver FIFO trigger points at 1, 4, 8, and 14 bytes.
- Built-in baud rate generator.

PIN DESCRIPTION

Table 57: UART 0 Pin Description

Pin Name	Type	Description
RxD0	Input	**Serial Input**. Serial receive data.
TxD0	Output	**Serial Output**. Serial transmit data.

REGISTER DESCRIPTION

Table 58: UART 0 Register Map

Address Offset	Name	Description	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	Access	Reset Value*
0xE000C000 DLAB = 0	U0RBR	Receiver Buffer Register	MSB			READ DATA				LSB	RO	un-defined
0xE000C000 DLAB = 0	U0THR	Transmit Holding Register	MSB			WRITE DATA				LSB	WO	NA
0xE000C004 DLAB = 0	U0IER	Interrupt Enable Register	0	0	0	0	0	Enable Rx Line Status Interrupt	Enable THRE Interrupt	Enable Rx Data Available Interrupt	R/W	0
0xE000C008	U0IIR	Interrupt ID Register	FIFOs Enabled		0	0	IIR3	IIR2	IIR1	IIR0	RO	0x01
0xE000C008	U0FCR	FIFO Control Register	Rx Trigger		Reserved		-	Tx FIFO Reset	Rx FIFO Reset	FIFO Enable	WO	0
0xE000C00C	U0LCR	Line Control Register	DLAB	Set Break	Stick Parity	Even Parity Select	Parity Enable	Number of Stop Bits	Word Length Select		R/W	0
0xE000C014	U0LSR	Line Status Register	Rx FIFO Error	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60
0xE000C01C	U0SCR	Scratch Pad Register	MSB							LSB	R/W	0
0xE000C000 DLAB = 1	U0DLL	Divisor Latch LSB	MSB							LSB	R/W	0x01
0xE000C004 DLAB = 1	U0DLM	Divisor Latch MSB	MSB							LSB	R/W	0

*Reset Value refers to the data stored in used bits only. It does not include reserved bits content.

UART 0 contains ten 8-bit registers as shown in Table 58. The Divisor Latch Access Bit (DLAB) is contained in U0LCR7 and enables access to the Divisor Latches.

UART 0 Receiver Buffer Register (U0RBR - 0xE000C000 when DLAB = 0, Read Only)

The U0RBR is the top byte of the UART0 Rx FIFO. The top byte of the Rx FIFO contains the oldest character received and can be read via the bus interface. The LSB (bit 0) represents the “oldest” received data bit. If the character received is less than 8 bits, the unused MSBs are padded with zeroes.

The Divisor Latch Access Bit (DLAB) in U0LCR must be zero in order to access the U0RBR. The U0RBR is always Read Only.

Table 59: UART0 Receiver Buffer Register (U0RBR - 0xE000C000 when DLAB = 0, Read Only)

U0RBR	Function	Description	Reset Value
7:0	Receiver Buffer Register	The UART0 Receiver Buffer Register contains the oldest received byte in the UART0 Rx FIFO.	un-defined

UART0 Transmitter Holding Register (U0THR - 0xE000C000 when DLAB = 0, Write Only)

The U0THR is the top byte of the UART0 Tx FIFO. The top byte is the newest character in the Tx FIFO and can be written via the bus interface. The LSB represents the first bit to transmit.

The Divisor Latch Access Bit (DLAB) in U0LCR must be zero in order to access the U0THR. The U0THR is always Write Only.

Table 60: UART0 Transmit Holding Register (U0THR - 0xE000C000 when DLAB = 0, Write Only)

U0THR	Function	Description	Reset Value
7:0	Transmit Holding Register	Writing to the UART0 Transmit Holding Register causes the data to be stored in the UART0 transmit FIFO. The byte will be sent when it reaches the bottom of the FIFO and the transmitter is available.	N/A

UART0 Divisor Latch LSB Register (U0DLL - 0xE000C000 when DLAB = 1)

UART0 Divisor Latch MSB Register (U0DLM - 0xE000C004 when DLAB = 1)

The UART0 Divisor Latch is part of the UART0 Baud Rate Generator and holds the value used to divide the VPB clock (pclk) in order to produce the baud rate clock, which must be 16x the desired baud rate. The U0DLL and U0DLM registers together form a 16 bit divisor where U0DLL contains the lower 8 bits of the divisor and U0DLM contains the higher 8 bits of the divisor. A ‘h0000 value is treated like a ‘h0001 value as division by zero is not allowed.The Divisor Latch Access Bit (DLAB) in U0LCR must be one in order to access the UART0 Divisor Latches.

Table 61: UART0 Divisor Latch LSB Register (U0DLL - 0xE000C000 when DLAB = 1)

U0DLL	Function	Description	Reset Value
7:0	Divisor Latch LSB Register	The UART0 Divisor Latch LSB Register, along with the U0DLM register, determines the baud rate of the UART0.	0x01

Table 62: UART0 Divisor Latch MSB Register (U0DLM - 0xE000C004 when DLAB = 1)

U0DLM	Function	Description	Reset Value
7:0	Divisor Latch MSB Register	The UART0 Divisor Latch MSB Register, along with the U0DLL register, determines the baud rate of the UART0.	0

UART0 Interrupt Enable Register (U0IER - 0xE000C004 when DLAB = 0)

The U0IER is used to enable the four UART0 interrupt sources.

Table 63: UART0 Interrupt Enable Register Bit Descriptions (U0IER - 0xE000C004 when DLAB = 0)

U0IER	Function	Description	Reset Value
0	RBR Interrupt Enable	0: Disable the RDA interrupt. 1: Enable the RDA interrupt. U0IER0 enables the Receive Data Available interrupt for UART0. It also controls the Character Receive Time-out interrupt.	0
1	THRE Interrupt Enable	0: Disable the THRE interrupt. 1: Enable the THRE interrupt. U0IER1 enables the THRE interrupt for UART0. The status of this interrupt can be read from U0LSR5.	0
2	Rx Line Status Interrupt Enable	0: Disable the Rx line status interrupts. 1: Enable the Rx line status interrupts. U0IER2 enables the UART0 Rx line status interrupts. The status of this interrupt can be read from U0LSR[4:1].	0
7:3	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

UART0 Interrupt Identification Register (U0IIR - 0xE000C008, Read Only)

The U0IIR provides a status code that denotes the priority and source of a pending interrupt. The interrupts are frozen during an U0IIR access. If an interrupt occurs during an U0IIR access, the interrupt is recorded for the next U0IIR access.

Table 64: UART0 Interrupt Identification Register Bit Descriptions (U0IIR - 0xE000C008, Read Only)

U0IIR	Function	Description	Reset Value
0	Interrupt Pending	0: At least one interrupt is pending. 1: No pending interrupts. Note that U0IIR0 is active low. The pending interrupt can be determined by evaluating U0IER3:1.	1
3:1	Interrupt Identification	011: 1. Receive Line Status (RLS) 010: 2a.Receive Data Available (RDA) 110: 2b.Character Time-out Indicator (CTI) 001: 3. THRE Interrupt. U0IER3 identifies an interrupt corresponding to the UART0 Rx FIFO. All other combinations of U0IER3:1 not listed above are reserved (000,100,101,111).	0
5:4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	FIFO Enable	These bits are equivalent to U0FCR0.	0

Interrupts are handled as described in Table 65. Given the status of U0IIR[3:0], an interrupt handler routine can determine the cause of the interrupt and how to clear the active interrupt. Interrupts are handled as described in Table 65. The U0IIR must be read in order to clear the interrupt prior to exitting the Interrupt Service Routine.

The UART0 RLS interrupt (U0IIR3:1=011) is the highest priority interrupt and is set whenever any one of four error conditions occur on the UART0 Rx input: overrun error (OE), parity error (PE), framing error (FE) and break interrupt (BI). The UART0 Rx error condition that set the interrupt can be observed via U0LSR4:1. The interrupt is cleared upon an U0LSR read.

The UART0 RDA interrupt (U0IIR3:1=010) shares the second level priority with the CTI interrupt (U0IIR3:1=110). The RDA is activated when the UART0 Rx FIFO reaches the trigger level defined in U0FCR7:6 and is reset when the UART0 Rx FIFO depth falls below the trigger level. When the RDA interrupt goes active, the CPU can read a block of data defined by the trigger level.

The CTI interrupt (U0IIR3:1=110) is a second level interrupt and is set when the UART0 Rx FIFO contains at least one character and no UART0 Rx FIFO activity has occurred in 3.5 to 4.5 character times. Any UART0 Rx FIFO activity (read or write of UART0 RSR) will clear the interrupt. This interrupt is intended to flush the UART0 RBR after a message has been received that is not a multiple of the trigger level size. For example, if a peripheral wished to send a 105 character message and the trigger level was 10 characters, the CPU would receive 10 RDA interrupts resulting in the transfer of 100 characters and 1 to 5 CTI interrupts (depending on the service routine) resulting in the transfer of the remaining 5 characters.

Table 65: UART0 Interrupt Handling

U0IIR[3:0]	Priority	Interrupt Type	Interrupt Source	Interrupt Reset
0001	-	none	none	-
0110	Highest	Rx Line Status / Error	OE or PE or FE or BI	U0LSR Read
0100	Second	Rx Data Available	Rx data available or trigger level reached in FIFO (U0FCR0=1)	U0RBR Read or UART0 FIFO drops below trigger level
1100	Second	Character Time-out Indication	Minimum of one character in the Rx FIFO and no character input or removed during a time period depending on how many characters are in FIFO and what the trigger level is set at (3.5 to 4.5 character times). The exact time will be: [(word length) X 7 - 2] X 8 + {(trigger level - number of characters) X 8 + 1] RCLKs	U0 RBR Read
0010	Third	THRE	THRE	U0IIR Read (if source of interrupt) or THR write
note: values "0000", “0011”, “0101”, “0111”, “1000”, “1001”, “1010”, “1011”,”1101”,”1110”,”1111” are reserved.				

The UART0 THRE interrupt (U0IIR3:1=001) is a third level interrupt and is activated when the UART0 THR FIFO is empty provided certain initialization conditions have been met. These initialization conditions are intended to give the UART0 THR FIFO a chance to fill up with data to eliminate many THRE interrupts from occurring at system start-up. The initialization conditions implement a one character delay minus the stop bit whenever THRE=1 and there have not been at least two characters in the U0THR at one time since the last THRE=1 event. This delay is provided to give the CPU time to write data to U0THR without a THRE interrupt to decode and service. A THRE interrupt is set immediately if the UART0 THR FIFO has held two or more characters at one time and currently, the U0THR is empty. The THRE interrupt is reset when a U0THR write occurs or a read of the U0IIR occurs and the THRE is the highest interrupt (U0IIR3:1=001).

UART0 FIFO Control Register (U0FCR - 0xE000C008)

The U0FCR controls the operation of the UART0 Rx and Tx FIFOs.

Table 66: UART0 FIFO Control Register Bit Descriptions (U0FCR - 0xE000C008)

U0FCR	Function	Description	Reset Value
0	FIFO Enable	Active high enable for both UART0 Rx and Tx FIFOs and U0FCR7:1 access. This bit must be set for proper UART0 opearation. Any transition on this bit will automatically clear the UART0 FIFOs.	0
1	Rx FIFO Reset	Writing a logic 1 to U0FCR1 will clear all bytes in UART0 Rx FIFO and reset the pointer logic. This bit is self-clearing.	0
2	Tx FIFO Reset	Writing a logic 1 to U0FCR2 will clear all bytes in UART0 Tx FIFO and reset the pointer logic. This bit is self-clearing.	0
5:3	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	Rx Trigger Level Select	00: trigger level 0 (default='h1) 01: trigger level 1 (default='h4) 10: trigger level 2 (default='h8) 11: trigger level 3 (default='he) These two bits determine how many receiver UART0 FIFO characters must be written before an interrupt is activated. The four trigger levels are defined by the user at compilation allowing the user to tune the trigger levels to the FIFO depths chosen.	0

UART0 Line Control Register (U0LCR - 0xE000C00C)

The U0LCR determines the format of the data character that is to be transmitted or received.

Table 67: UART0 Line Control Register Bit Descriptions (U0LCR - 0xE000C00C)

U0LCR	Function	Description	Reset Value
1:0	Word Length Select	00: 5 bit character length 01: 6 bit character length 10: 7 bit character length 11: 8 bit character length	0
2	Stop Bit Select	0: 1 stop bit 1: 2 stop bits (1.5 if U0LCR[1:0]=00)	0
3	Parity Enable	0: Disable parity generation and checking 1: Enable parity generation and checking	0
5:4	Parity Select	00: Odd parity 01: Even parity 10: Forced “1” stick parity 11: Forced “0” stick parity	0
6	Break Control	0: Disable break transmission 1: Enable break transmission. Output pin UART0 TxD is forced to logic 0 when U0LCR6 is active high.	0
7	Divisor Latch Access Bit	0: Disable access to Divisor Latches 1: Enable access to Divisor Latches	0

UART0 Line Status Register (U0LSR - 0xE000C014, Read Only)

The U0LSR is a read-only register that provides status information on the UART0 Tx and Rx blocks.

Table 68: UART0 Line Status Register Bit Descriptions (U0LSR - 0xE000C014, Read Only)

U0LSR	Function	Description	Reset Value
0	Receiver Data Ready (RDR)	0: U0RBR is empty 1: U0RBR contains valid data U0LSR0 is set when the U0RBR holds an unread character and is cleared when the UART0 RBR FIFO is empty.	0
1	Overrun Error (OE)	0: Overrun error status is inactive. 1: Overrun error status is active. The overrun error condition is set as soon as it occurs. An U0LSR read clears U0LSR1. U0LSR1 is set when UART0 RSR has a new character assembled and the UART0 RBR FIFO is full. In this case, the UART0 RBR FIFO will not be overwritten and the character in the UART0 RSR will be lost.	0
2	Parity Error (PE)	0: Parity error status is inactive. 1: Parity error status is active. When the parity bit of a received character is in the wrong state, a parity error occurs. An U0LSR read clears U0LSR2. Time of parity error detection is dependent on U0FCR0. A parity error is associated with the character being read from the UART0 RBR FIFO.	0
3	Framing Error (FE)	0: Framing error status is inactive. 1: Framing error status is active. When the stop bit of a received character is a logic 0, a framing error occurs. An U0LSR read clears U0LSR3. The time of the framing error detection is dependent on U0FCR0. A framing error is associated with the character being read from the UART0 RBR FIFO. Upon detection of a framing error, the Rx will attempt to resynchronize to the data and assume that the bad stop bit is actually an early start bit. However, it cannot be assumed that the next received byte will be correct even if there is no Framing Error.	0
4	Break Interrupt (BI)	0: Break interrupt status is inactive. 1: Break interrupt status is active. When RxD0 is held in the spacing state (all 0's) for one full character transmission (start, data, parity, stop), a break interrupt occurs. Once the break condition has been detected, the receiver goes idle until RxD0 goes to marking state (all 1's). An U0LSR read clears this status bit. The time of break detection is dependent on U0FCR0. The break interrupt is associated with the character being read from the UART0 RBR FIFO.	0
5	Transmitter Holding Register Empty (THRE)	0: U0THR contains valid data. 1: U0THR is empty. THRE is set immediately upon detection of an empty UART0 THR and is cleared on a U0THR write.	1
6	Transmitter Empty (TEMT)	0: U0THR and/or the U0TSR contains valid data. 1: U0THR and the U0TSR are empty. TEMT is set when both U0THR and U0TSR are empty; TEMT is cleared when either the U0TSR or the U0THR contain valid data.	1
7	Error in Rx FIFO (RXFE)	0: U0RBR contains no UART0 Rx errors or U0FCR0=0. 1: UART0 RBR contains at least one UART0 Rx error. U0LSR7 is set when a character with a Rx error such as framing error, parity error or break interrupt, is loaded into the U0RBR. This bit is cleared when the U0LSR register is read and there are no subsequent errors in the UART0 FIFO.	0

UART0 Scratch Pad Register (U0SCR - 0xE000C01C)

The U0SCR has no effect on the UART0 operation. This register can be written and/or read at user's discretion. There is no provision in the interrupt interface that would indicate to the host that a read or write of the U0SCR has occurred.

Table 69: UART0 Scratchpad Register (U0SCR - 0xE000C01C)

U0SCR	Function	Description	Reset Value
7:0	-	A readable, writable byte.	0

ARCHITECTURE

The architecture of the UART0 is shown below in the block diagram.

The VPB interface provides a communications link between the CPU or host and the UART0.

The UART0 receiver block, U0Rx, monitors the serial input line, RxD0, for valid input. The UART0 Rx Shift Register (U0RSR) accepts valid characters via RxD0. After a valid character is assembled in the U0RSR, it is passed to the UART0 Rx Buffer Register FIFO to await access by the CPU or host via the generic host interface.

The UART0 transmitter block, U0Tx, accepts data written by the CPU or host and buffers the data in the UART0 Tx Holding Register FIFO (U0THR). The UART0 Tx Shift Register (U0TSR) reads the data stored in the U0THR and assembles the data to transmit via the serial output pin, TxD0.

The UART0 Baud Rate Generator block, U0BRG, generates the timing enables used by the UART0 Tx block. The U0BRG clock input source is the VPB clock (pclk). The main clock is divided down per the divisor specified in the U0DLL and U0DLM registers. This divided down clock is a 16x oversample clock, NBAUDOUT.

The interrupt interface contains registers U0IER and U0IIR. The interrupt interface receives several one clock wide enables from the U0Tx and U0Rx blocks.

Status information from the U0Tx and U0Rx is stored in the U0LSR. Control information for the U0Tx and U0Rx is stored in the U0LCR.

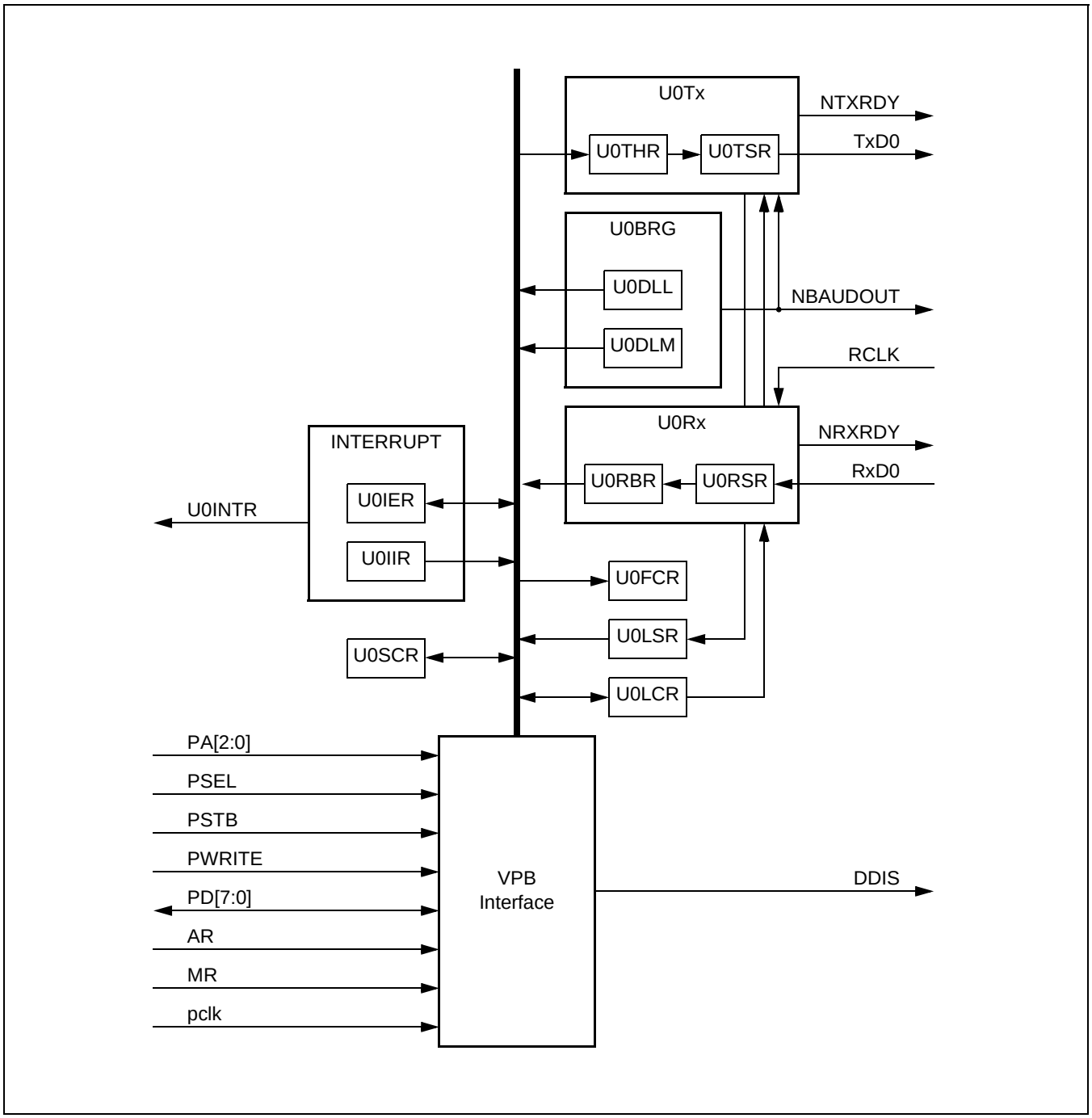


Figure 15: UART0 Block Diagram

10. UART 1

FEATURES

- UART 1 is identical to UART 0, with the addition of a modem interface.
- 16 byte Receive and Transmit FIFOs.
- Register locations conform to ‘550 industry standard.
- Receiver FIFO trigger points at 1, 4, 8, and 14 bytes.
- Built-in baud rate generator.
- Standard modem interface signals included.

PIN DESCRIPTION

Table 70: UART1 Pin Description

Pin Name	Type	Description
RxD1	Input	**Serial Input**. Serial receive data.
TxD1	Output	**Serial Output**. Serial transmit data.
CTS1	Input	**Clear To Send**. Active low signal indicates if the external modem is ready to accept transmitted data via TxD1 from the UART1. In normal operation of the modem interface (U1MCR4=0), the complement value of this signal is stored in U1MSR4. State change information is stored in U1MSR0 and is a source for a priority level 4 interrupt, if enabled (U1IER3=1).
DCD1	Input	**Data Carrier Detect**. Active low signal indicates if the external modem has established a communication link with the UART1 and data may be exchanged. In normal operation of the modem interface (U1MCR4=0), the complement value of this signal is stored in U1MSR7. State change information is stored in U1MSR3 and is a source for a priority level 4 interrupt, if enabled (U1IER3=1).
DSR1	Input	**Data Set Ready**. Active low signal indicates if the external modem is ready to establish a communications link with the UART1. In normal operation of the modem interface (U1MCR4=0), the complement value of this signal is stored in U1MSR5. State change information is stored in U1MSR1 and is a source for a priority level 4 interrupt, if enabled (U1IER3=1).
DTR1	Output	**Data Terminal Ready**. Active low signal indicates that the UART1 is ready to establish connection with external modem. The complement value of this signal is stored in U1MCR0.
RI1	Input	**Ring Indicator**. Active low signal indicates that a telephone ringing signal has been detected by the modem. In normal operation of the modem interface (U1MCR4=0), the complement value of this signal is stored in U1MSR6. State change information is stored in U1MSR2 and is a source for a priority level 4 interrupt, if enabled (U1IER3=1).
RTS1	Output	**Request To Send**. Active low signal indicates that the UART1 would like to transmit data to the external modem. The complement value of this signal is stored in U1MCR1.

REGISTER DESCRIPTION

Table 71: UART 1 Register Map

Address	Name	Description	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	Access	Reset Value*
0xE0010000 DLAB = 0	U1RBR	Receiver Buffer Register	MSB			READ DATA				LSB	RO	un-defined
0xE0010000 DLAB = 0	U1THR	Transmit Holding Register	MSB			WRITE DATA				LSB	WO	NA
0xE0010004 DLAB = 0	U1IER	Interrupt Enable Register	0	0	0	0	Enable Modem Status Interrupt	Enable Rx Line Status Interrupt	Enable THRE Interrupt	Enable Rx Data Available Interrupt	R/W	0
0xE0010008	U1IIR	Interrupt ID Register	FIFOs Enabled		0	0	IIR3	IIR2	IIR1	IIR0	RO	0x01
0xE0010008	U1FCR	FIFO Control Register	Rx Trigger		Reserved		-	Tx FIFO Reset	Rx FIFO Reset	FIFO Enable	WO	0
0xE001000C	U1LCR	Line Control Register	DLAB	Set Break	Stick Parity	Even Parity Select	Parity Enable	Number of Stop Bits	Word Length Select		R/W	0
0xE0010010	U1MCR	Modem Control Register	0	0	0	Loop Back	0	0	RTS	DTR	R/W	0
0xE0010014	U1LSR	Line Status Register	Rx FIFO Error	TEMT	THRE	BI	FE	PE	OE	DR	RO	0x60
0xE0010018	U1MSR	Modem Status Register	DCD	RI	DSR	CTS	Delta DCD	Trailing Edge RI	Delta DSR	Delta CTS	RO	0
0xE001001C	U1SCR	Scratch Pad Register	MSB							LSB	R/W	0
0xE0010000 DLAB = 1	U1DLL	Divisor Latch LSB	MSB							LSB	R/W	0
0xE0010004 DLAB = 1	U1DLM	Divisor Latch MSB	MSB							LSB	R/W	0

*Reset Value refers to the data stored in used bits only. It does not include reserved bits content.

UART1 contains twelve 8-bit registers as shown in Table 71. The Divisor Latch Access Bit (DLAB) is contained in U1LCR7 and enables access to the Divisor Latches.

UART1 Receiver Buffer Register (U1RBR - 0xE0010000 when DLAB = 0, Read Only)

The U1RBR is the top byte of the UART1 Rx FIFO. The top byte of the Rx FIFO contains the oldest character received and can be read via the bus interface. The LSB (bit 0) represents the “oldest” received data bit. If the character received is less than 8 bits, the unused MSBs are padded with zeroes.

The Divisor Latch Access Bit (DLAB) in U1LCR must be zero in order to access the U1RBR. The U1RBR is always Read Only.

Table 72: UART1 Receiver Buffer Register (U1RBR - 0xE0010000 when DLAB = 0, Read Only)

U1RBR	Function	Description	Reset Value
7:0	Receiver Buffer Register	The UART1 Receiver Buffer Register contains the oldest received byte in the UART1 Rx FIFO.	un-defined

UART1 Transmitter Holding Register (U1THR - 0xE0010000 when DLAB = 0, Write Only)

The U1THR is the top byte of the UART1 Tx FIFO. The top byte is the newest character in the Tx FIFO and can be written via the bus interface. The LSB represents the first bit to transmit.

The Divisor Latch Access Bit (DLAB) in U1LCR must be zero in order to access the U1THR. The U1THR is always Write Only.

Table 73: UART1 Transmit Holding Register (U1THR - 0xE0010000 when DLAB = 0, Write Only)

U1THR	Function	Description	Reset Value
7:0	Transmit Holding Register	Writing to the UART1 Transmit Holding Register causes the data to be stored in the UART1 transmit FIFO. The byte will be sent when it reaches the bottom of the FIFO and the transmitter is available.	N/A

UART1 Divisor Latch LSB Register (U1DLL - 0xE0010000 when DLAB = 1)

UART1 Divisor Latch MSB Register (U1DLM - 0xE0010004 when DLAB = 1)

The UART1 Divisor Latch is part of the UART1 Baud Rate Generator and holds the value used to divide the VPB clock (pclk) in order to produce the baud rate clock, which must be 16x the desired baud rate. The U1DLL and U1DLM registers together form a 16 bit divisor where U1DLL contains the lower 8 bits of the divisor and U1DLM contains the higher 8 bits of the divisor. A ‘h0000 value is treated like a ‘h0001 value as division by zero is not allowed.The Divisor Latch Access Bit (DLAB) in U1LCR must be one in order to access the UART1 Divisor Latches.

Table 74: UART1 Divisor Latch LSB Register (U1DLL - 0xE0010000 when DLAB = 1)

U1DLL	Function	Description	Reset Value
7:0	Divisor Latch LSB Register	The UART1 Divisor Latch LSB Register, along with the U1DLM register, determines the baud rate of the UART1.	0x01

Table 75: UART1 Divisor Latch MSB Register (U1DLM - 0xE0010004 when DLAB = 1)

U1DLM	Function	Description	Reset Value
7:0	Divisor Latch MSB Register	The UART1 Divisor Latch MSB Register, along with the U1DLL register, determines the baud rate of the UART1.	0

UART1 Interrupt Enable Register (U1IER - 0xE0010004 when DLAB = 0)

The U1IER is used to enable the four interrupt sources.

Table 76: UART1 Interrupt Enable Register Bit Descriptions (U1IER - 0xE0010004 when DLAB = 0)

U1IER	Function	Description	Reset Value
0	RBR Interrupt Enable	0: Disable the RDA interrupt. 1: Enable the RDA interrupt. U1IER0 enables the Receive Data Available interrupt for UART1. It also controls the Receive Time-out interrupt.	0
1	THRE Interrupt Enable	0: Disable the THRE interrupt. 1: Enable the THRE interrupt. U1IER1 enables the THRE interrupt for UART1. The status of this interrupt can be read from U1LSR5.	0
2	Rx Line Status Interrupt Enable	0: Disable the Rx line status interrupts. 1: Enable the Rx line status interrupts. U1IER2 enables the UART1 Rx line status interrupts. The status of this interrupt can be read from U1LSR[4:1].	0
3	Modem Status Interrupt Enable	0: Disable the modem interrupt. 1: Enable the modem interrupt. U1IER3 enables the modem interrupt. The status of this interrupt can be read from U1MSR[3:0].	0
7:4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

UART1 Interrupt Identification Register (U1IIR - 0xE0010008, Read Only)

The U1IIR provides a status code that denotes the priority and source of a pending interrupt. The interrupts are frozen during an U1IIR access. If an interrupt occurs during an U1IIR access, the interrupt is recorded for the next U1IIR access.

Table 77: UART1 Interrupt Identification Register Bit Descriptions (IIR - 0xE0010008, Read Only)

U1IIR	Function	Description	Reset Value
0	Interrupt Pending	0: At least one interrupt is pending. 1: No pending interrupts. Note that U1IIR0 is active low. The pending interrupt can be determined by evaluating U1IIR3:1.	1
3:1	Interrupt Identification	011: 1. Receive Line Status (RLS) 010: 2a.Receive Data Available (RDA) 110: 2b.Character Time-out Indicator (CTI) 001: 3. THRE Interrupt. 000: 4. Modem Interrupt. U1IER3 identifies an interrupt corresponding to the UART1 Rx FIFO and modem signals. All other combinations of U1IER3:1 not listed above are reserved (100,101,111).	0
5:4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	FIFO Enable	These bits are equivalent to U1FCR0.	0

Interrupts are handled as described in Table 78. Given the status of U1IIR[3:0], an interrupt handler routine can determine the cause of the interrupt and how to clear the active interrupt. The U1IIR must be read in order to clear the interrupt prior to exiting the Interrupt Service Routine.

The UART1 RLS interrupt (U1IIR3:1=011) is the highest priority interrupt and is set whenever any one of four error conditions occur on the UART1Rx input: overrun error (OE), parity error (PE), framing error (FE) and break interrupt (BI). The UART1 Rx error condition that set the interrupt can be observed via U1LSR4:1. The interrupt is cleared upon an U1LSR read.

The UART1 RDA interrupt (U1IIR3:1=010) shares the second level priority with the CTI interrupt (U1IIR3:1=110). The RDA is activated when the UART1 Rx FIFO reaches the trigger level defined in U1FCR7:6 and is reset when the UART1 Rx FIFO depth falls below the trigger level. When the RDA interrupt goes active, the CPU can read a block of data defined by the trigger level.

The CTI interrupt (U1IIR3:1=110) is a second level interrupt and is set when the UART1 Rx FIFO contains at least one character and no UART1 Rx FIFO activity has occurred in 3.5 to 4.5 character times. Any UART1 Rx FIFO activity (read or write of UART1 RSR) will clear the interrupt. This interrupt is intended to flush the UART1 RBR after a message has been received that is not a multiple of the trigger level size. For example, if a peripheral wished to send a 105 character message and the trigger level was 10 characters, the CPU would receive 10 RDA interrupts resulting in the transfer of 100 characters and 1 to 5 CTI interrupts (depending on the service routine) resulting in the transfer of the remaining 5 characters.

Table 78: UART1 Interrupt Handling

U1IIR[3:0]	Priority	Interrupt Type	Interrupt Source	Interrupt Reset
0001	-	none	none	-
0110	Highest	Rx Line Status / Error	OE or PE or FE or BI	U1LSR Read
0100	Second	Rx Data Available	Rx data available or trigger level reached in FIFO mode (FCR0=1)	U1RBR Read or UART1 FIFO drops below trigger level
1100	Second	Character Time-out Indication	Minimum of one character in the Rx FIFO and no character input or removed during a time period depending on how many characters are in FIFO and what the trigger level is set at (3.5 to 4.5 character times). The exact time will be: [(word length) X 7 - 2] X 8 + {(trigger level - number of characters) X 8 + 1] RCLKs	U1RBR Read
0010	Third	THRE	THRE	U1IIR Read (if source of interrupt) or THR write
0000	Fourth	Modem Status	CTS or DSR or RI or DCD	MSR Read
note: values “0011”. “0101”, “0111”, “1000”, “1001”, “1010”, “1011”,”1101”,”1110”,”1111” are reserved.				

The UART1 THRE interrupt (U1IIR3:1=001) is a third level interrupt and is activated when the UART1 THR FIFO is empty provided certain initialization conditions have been met. These initialization conditions are intended to give the UART1 THR FIFO a chance to fill up with data to eliminate many THRE interrupts from occurring at system start-up. The initialization conditions implement a one character delay minus the stop bit whenever THRE=1 and there have not been at least two characters in the U1THR at one time since the last THRE=1 event. This delay is provided to give the CPU time to write data to U1THR without a THRE interrupt to decode and service. A THRE interrupt is set immediately if the UART1 THR FIFO has held two or more characters at one time and currently, the U1THR is empty. The THRE interrupt is reset when a U1THR write occurs or a read of the U1IIR occurs and the THRE is the highest interrupt (U1IIR3:1=001).

The modem interrupt (U1IIR3:1=000) is the lowest priority interrupt and is activated whenever there is any state change on modem inputs pins, DCD, DSR or CTS. In addition, a low to high transition on modem input RI will generate a modem interrupt. The source of the modem interrupt can be determined by examining U1MSR3:0. A U1MSR read will clear the modem interrupt.

UART1 FIFO Control Register (U1FCR - 0xE0010008)

The U1FCR controls the operation of the UART1 Rx and Tx FIFOs.

Table 79: UART1 FCR Bit Descriptions (U1FCR - 0xE0010008)

U1FCR	Function	Description	Reset Value
0	FIFO Enable	Active high enable for both UART1 Rx and Tx FIFOs and U1FCR7:1 access. This bit must be set for proper UART1 operation. Any transition on this bit will automatically clear the UART1 FIFOs.	0
1	Rx FIFO Reset	Writing a logic 1 to U1FCR1 will clear all bytes in UART1 Rx FIFO and reset the pointer logic. This bit is self-clearing.	0
2	Tx FIFO Reset	Writing a logic 1 to U1FCR2 will clear all bytes in UART1 Tx FIFO and reset the pointer logic. This bit is self-clearing.	0
5:3	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	Rx Trigger Level Select	00: trigger level 0 (default='h1) 01: trigger level 1 (default='h4) 10: trigger level 2 (default='h8) 11: trigger level 3 (default='he) These two bits determine how many receiver UART1 FIFO characters must be written before an interrupt is activated. The four trigger levels are defined by the user at compilation allowing the user to tune the trigger levels to the FIFO depths chosen.	0

UART1 Line Control Register (U1LCR - 0xE001000C)

The U1LCR determines the format of the data character that is to be transmitted or received.

Table 80: UART1 Line Control Register Bit Descriptions (U1LCR - 0xE001000C)

U1LCR	Function	Description	Reset Value
1:0	Word Length Select	00: 5 bit character length 01: 6 bit character length 10: 7 bit character length 11: 8 bit character length	0
2	Stop Bit Select	0: 1 stop bit 1: 2 stop bits (1.5 if U1LCR[1:0]=00)	0
3	Parity Enable	0: Disable parity generation and checking 1: Enable parity generation and checking	0
5:4	Parity Select	00: Odd parity 01: Even parity 10: Forced “1” stick parity 11: Forced “0” stick parity	0
6	Break Control	0: Disable break transmission 1: Enable break transmission. Output pin UART1 TxD is forced to logic 0 when U1LCR6 is active high.	0
7	Divisor Latch Access Bit	0: Disable access to Divisor Latches 1: Enable access to Divisor Latches	0

UART1 Modem Control Register (U1MCR - 0xE0010010)

The U1MCR enables the modem loopback mode and controls the modem output signals.

Table 81: UART1 Modem Control Register Bit Descriptions (U1MCR - 0xE0010010)

U1MCR	Function	Description	Reset Value
0	DTR Control	Source for modem output pin, DTR. This bit reads as 0 when modem loopback mode is active.	0
1	RTS Control	Source for modem output pin RTS. This bit reads as 0 when modem loopback mode is active.	0
2	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
4	Loopback Mode Select	0: Disable modem loopback mode 1: Enable modem loopback mode The modem loopback mode provides a mechanism to perform diagnostic loopback testing. Serial data from the transmitter is connected internally to serial input of the receiver. Input pin, RxD1, has no effect on loopback and output pin, TxD1 is held in marking state. The four modem inputs (CTS, DSR, RI and DCD) are disconnected externally. Externally, the modem outputs (RTS, DTR) are set inactive. Internally, the four modem outputs are connected to the four modem inputs. As a result of these connections, the upper four bits of the U1MSR will be driven by the lower four bits of the U1MCR rather than the four modem inputs in normal mode. This permits modem status interrupts to be generated in loopback mode by writing the lower four bits of U1MCR.	0
7:5	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

UART1 Line Status Register (U1LSR - 0xE0010014, Read Only)

The U1LSR is a read-only register that provides status information on the UART1 Tx and Rx blocks.

Table 82: UART1 Line Status Register Bit Descriptions (U1LSR - 0xE0010014, Read Only)

U1LSR	Function	Description	Reset Value
0	Receiver Data Ready (RDR)	0: U1RBR is empty 1: U1RBR contains valid data U1LSR0 is set when the U1RBR holds an unread character and is cleared when the UART1 RBR FIFO is empty.	0
1	Overrun Error (OE)	0: Overrun error status is inactive. 1: Overrun error status is active. The overrun error condition is set as soon as it occurs. An U1LSR read clears U1LSR1. U1LSR1 is set when UART1 RSR has a new character assembled and the UART1 RBR FIFO is full. In this case, the UART1 RBR FIFO will not be overwritten and the character in the UART1 RSR will be lost.	0
2	Parity Error (PE)	0: Parity error status is inactive. 1: Parity error status is active. When the parity bit of a received character is in the wrong state, a parity error occurs. An U1LSR read clears U1LSR2. Time of parity error detection is dependent on U1FCR0. A parity error is associated with the character being read from the UART1 RBR FIFO.	0
3	Framing Error (FE)	0: Framing error status is inactive. 1: Framing error status is active. When the stop bit of a received character is a logic 0, a framing error occurs. An U1LSR read clears this bit. The time of the framing error detection is dependent on U1FCR0. A framing error is associated with the character being read from the UART1 RBR FIFO. Upon detection of a framing error, the Rx will attempt to resynchronize to the data and assume that the bad stop bit is actually an early start bit.	0
4	Break Interrupt (BI)	0: Break interrupt status is inactive. 1: Break interrupt status is active. When RxD1 is held in the spacing state (all 0's) for one full character transmission (start, data, parity, stop), a break interrupt occurs. Once the break condition has been detected, the receiver goes idle until RxD1 goes to marking state (all 1's). An U1LSR read clears this status bit. The time of break detection is dependent on U1FCR0. The break interrupt is associated with the character being read from the UART1 RBR FIFO.	0
5	Transmitter Holding Register Empty (THRE)	0: U1THR contains valid data. 1: U1THR is empty. THRE is set immediately upon detection of an empty U1THR and is cleared on a U1THR write.	1
6	Transmitter Empty (TEMT)	0: U1THR and/or the U1TSR contains valid data. 1: U1THR and the U1TSR are empty. TEMT is set when both THR and TSR are empty; TEMT is cleared when either the U1TSR or the U1THR contain valid data.	1
7	Error in Rx FIFO (RXFE)	0: U1RBR contains no UART1 Rx errors or U1FCR0=0. 1: U1RBR contains at least one UART1 Rx error. U1LSR7 is set when a character with a Rx error such as framing error, parity error or break interrupt, is loaded into the U1RBR. This bit is cleared when the U1LSR register is read and there are no subsequent errors in the UART1 FIFO.	0

UART1 Modem Status Register (U1MSR - 0x0xE0010018)

The U1MSR is a read-only register that provides status information on the modem input signals. U1MSR3:0 is cleared on U1MSR read. Note that modem signals have no direct affect on UART1 operation, they facilitate software implementation of modem signal operations.

Table 83: UART1 Modem Status Register Bit Descriptions (U1MSR - 0x0xE0010018)

U1MSR	Function	Description	Reset Value
0	Delta CTS	0: No change detected on modem input, CTS. 1: State change detected on modem input, CTS. Set upon state change of input CTS. Cleared on an U1MSR read.	0
1	Delta DSR	0: No change detected on modem input, DSR. 1: State change detected on modem input, DSR. Set upon state change of input DSR. Cleared on an U1MSR read.	0
2	Trailing Edge RI	0: No change detected on modem input, RI. 1: Low-to-high transition detected on RI. Set upon low to high transition of input RI. Cleared on an U1MSR read.	0
3	Delta DCD	0: No change detected on modem input, DCD. 1: State change detected on modem input, DCD. Set upon state change of input DCD. Cleared on an U1MSR read.	0
4	CTS	Clear To Send State. Complement of input signal CTS. This bit is connected to U1MCR[1] in modem loopback mode.	0
5	DSR	Data Set Ready State. Complement of input signal DSR. This bit is connected to U1MCR[0] in modem loopback mode.	0
6	RI	Ring Indicator State. Complement of input RI. This bit is connected to U1MCR[2] in modem loopback mode.	0
7	DCD	Data Carrier Detect State. Complement of input DCD. This bit is connected to U1MCR[3] in modem loopback mode.	0

UART1 Scratch Pad Register (U1SCR - 0xE001001C)

The U1SCR has no effect on the UART1 operation. This register can be written and/or read at user's discretion. There is no provision in the interrupt interface that would indicate to the host that a read or write of the U1SCR has occurred.

Table 84: UART1 Scratchpad Register (U1SCR - 0xE001001C)

U1SCR	Function	Description	Reset Value
7:0	-	A readable, writable byte.	0

ARCHITECTURE

The architecture of the UART1 is shown below in the block diagram.

The VPB interface provides a communications link between the CPU or host and the UART1.

The UART1 receiver block, U1Rx, monitors the serial input line, RxD1, for valid input. The UART1 Rx Shift Register (U1RSR) accepts valid characters via RxD1. After a valid character is assembled in the U1RSR, it is passed to the UART1 Rx Buffer Register FIFO to await access by the CPU or host via the generic host interface.

The UART1 transmitter block, U1Tx, accepts data written by the CPU or host and buffers the data in the UART1 Tx Holding Register FIFO (U1THR). The UART1 Tx Shift Register (U1TSR) reads the data stored in the U1THR and assembles the data to transmit via the serial output pin, TxD1.

The UART1 Baud Rate Generator block, U1BRG, generates the timing enables used by the UART1 Tx block. The U1BRG clock input source is the VPB clock (pclk). The main clock is divided down per the divisor specified in the U1DLL and u1DLM registers. This divided down clock is a 16x oversample clock, NBAUDOUT.

The modem interface contains registers U1MCR and U1MSR. This interface is responsible for handshaking between a modem peripheral and the UART1.

The interrupt interface contains registers U1IER and U1IIR. The interrupt interface receives several one clock wide enables from the U1Tx, U1Rx and modem blocks.

Status information from the U1Tx and U1Rx is stored in the U1LSR. Control information for the U1Tx and U1Rx is stored in the U1LCR.

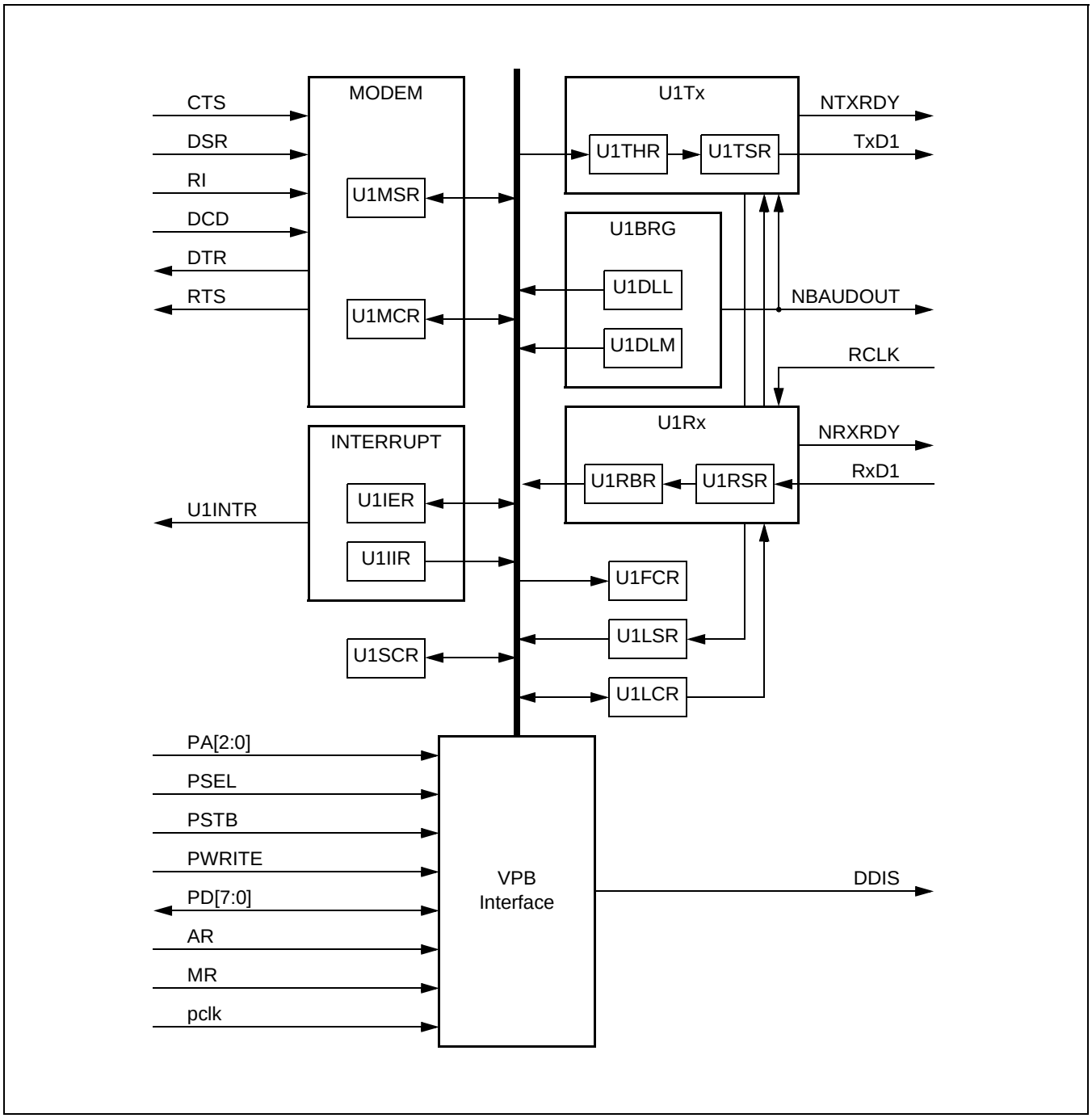


Figure 16: UART1 Block Diagram

11. I²C INTERFACE

FEATURES

- Standard I²C compliant bus interface.
- Easy to configure as Master, Slave, or Master/Slave.
- Programmable clocks allow versatile rate control.
- Bidirectional data transfer between masters and slaves.
- Multi-master bus (no central master).
- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus.
- Serial clock synchronization allows devices with different bit rates to communicate via one serial bus.
- Serial clock synchronization can be used as a handshake mechanism to suspend and resume serial transfer.
- The I²C bus may be used for test and diagnostic purposes.

APPLICATIONS

- Interfaces to external I²C standard parts, such as serial RAMs, LCDs, tone generators, etc.

DESCRIPTION

A typical I²C bus configuration is shown in Figure 17. Depending on the state of the direction bit (R/W), two types of data transfers are possible on the I²C bus:

- Data transfer from a master transmitter to a slave receiver. The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte.
- Data transfer from a slave transmitter to a master receiver. The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. Next follows the data bytes transmitted by the slave to the master. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a “not acknowledge” is returned. The master device generates all of the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a repeated START condition. Since a repeated START condition is also the beginning of the next serial transfer, the I²C bus will not be released.

This device provides a byte oriented I²C interface. It has four operating modes: master transmitter mode, master receiver mode, slave transmitter mode and slave receiver mode.

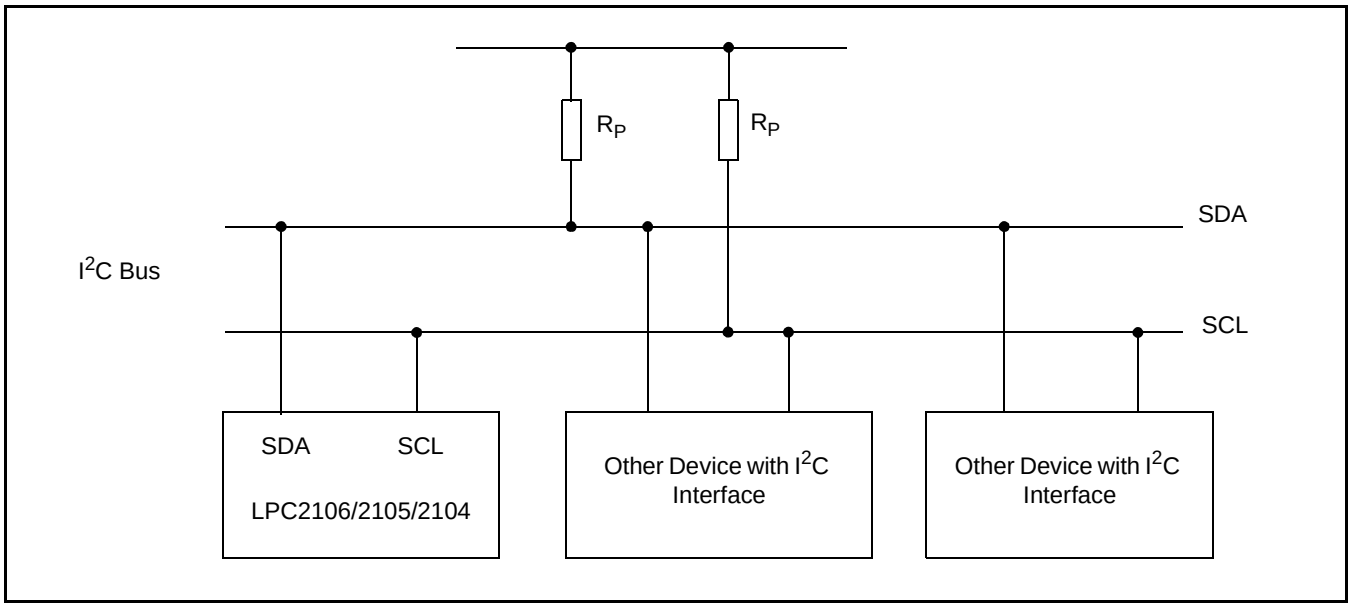


Figure 17: I^2C Bus Configuration

I^2C Operating Modes

Master Transmitter Mode:

In this mode data is transmitted from master to slave. Before the master transmitter mode can be entered, I2CONSET must be initialized as shown in Figure 18. I2EN must be set to 1 to enable the I^2C function. If the AA bit is 0, the I^2C interface will not acknowledge any address when another device is master of the bus, so it can not enter slave mode. The STA, STO and SI bits must be 0. The SI Bit is cleared by writing 1 to the SIC bit in the I2CONCLR register.

	7	6	5	4	3	2	1	0
I2CONSET	-	I2EN	STA	STO	SI	AA	-	-
	-	1	0	0	0	0	-	-

Figure 18: Slave Mode Configuration

The first byte transmitted contains the slave address of the receiving device (7 bits) and the data direction bit. In this mode the data direction bit (R/W) should be 0 which means Write. The first byte transmitted contains the slave address and Write bit. Data is transmitted 8 bits at a time. After each byte is transmitted, an acknowledge bit is received. START and STOP conditions are output to indicate the beginning and the end of a serial transfer.

The I^2C interface will enter master transmitter mode when software sets the STA bit. The I^2C logic will send the START condition as soon as the bus is free. After the START condition is transmitted, the SI bit is set, and the status code in I2STAT should be 08h. This status code must be used to vector to an interrupt service routine which should load the slave address and Write bit to I2DAT (Data Register), and then clear the SI bit. SI is cleared by writing a 1 to the SIC bit in the I2CONCLR register.

When the slave address and R/W bit have been transmitted and an acknowledgment bit has been received, the SI bit is set again, and the possible status codes now are 18h, 20h, or 38h for the master mode, or 68h, 78h, or 0B0h if the slave mode was enabled (by setting AA=1). The appropriate actions to be taken for each of these status codes are shown in Table 3 to Table 6 in "80C51 Family Derivatives 8XC552/562 Overview" datasheet available on-line at

http://www.semiconductors.philips.com/acrobat/various/8XC552_562OVERVIEW_2.pdf.

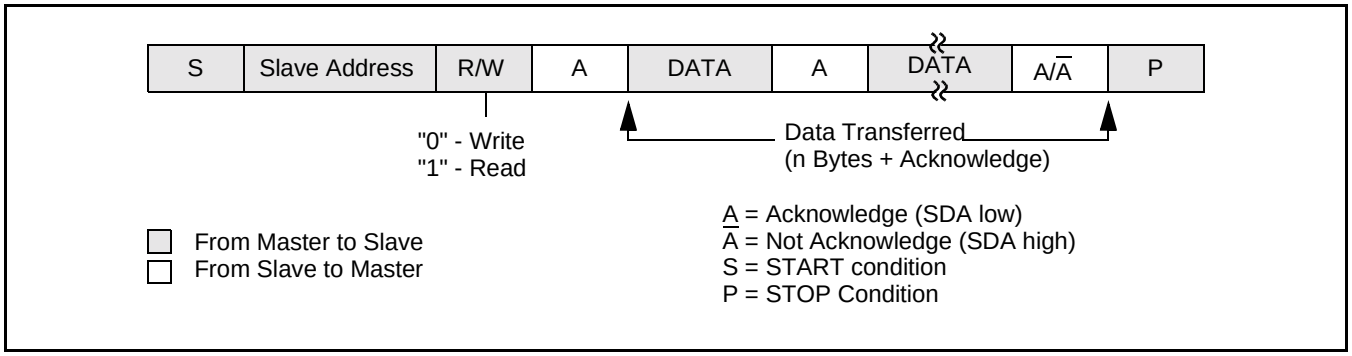


Figure 19: Format in the master transmitter mode

Master Receiver Mode:

In the master receiver mode, data is received from a slave transmitter. The transfer is initiated in the same way as in the master transmitter mode. When the START condition has been transmitted, the interrupt service routine must load the slave address and the data direction bit to I^2C Data Register (I2DAT), and then clear the SI bit.

When the slave address and data direction bit have been transmitted and an acknowledge bit has been received, the SI bit is set, and the Status Register will show the status code. For master mode, the possible status codes are 40H, 48H, or 38H. For slave mode, the possible status codes are 68H, 78H, or B0H. Refer to Table 4 in "80C51 Family Derivatives 8XC552/562 Overview" datasheet available on-line at

http://www.semiconductors.philips.com/acrobat/various/8XC552_562OVERVIEW_2.pdf

for details.

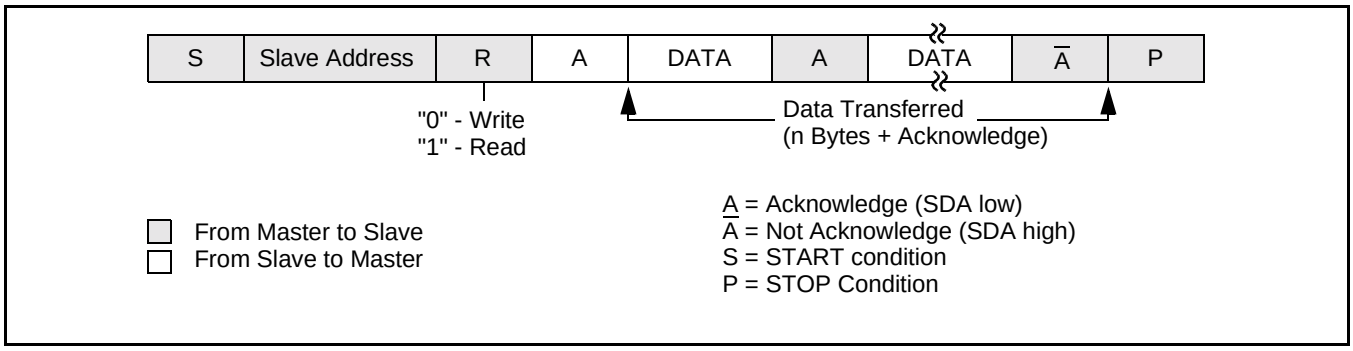


Figure 20: Format of master receiver mode

After a repeated START condition, I^2C may switch to the master transmitter mode.

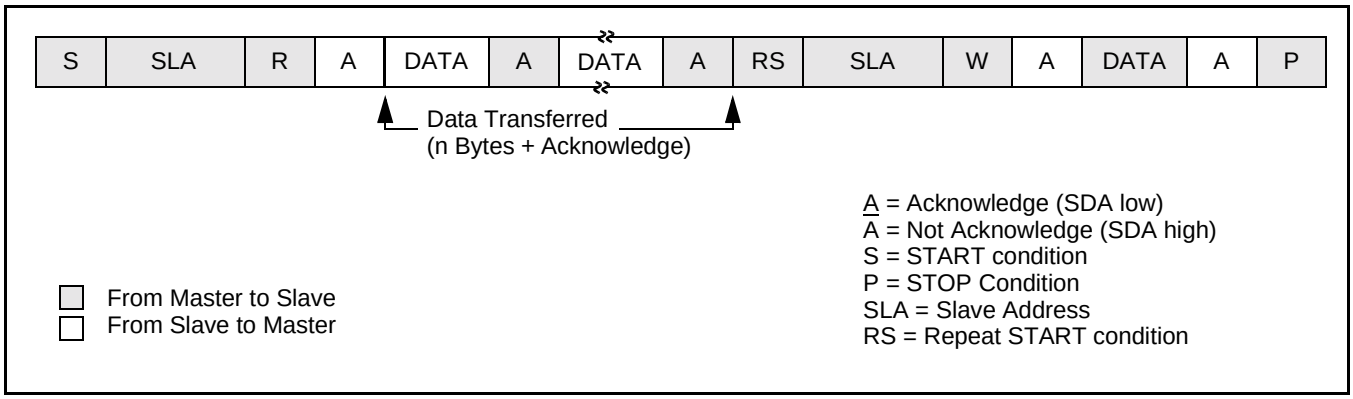


Figure 21: A master receiver switch to master transmitter after sending repeated START

Slave Receiver Mode:

In the slave receiver mode, data bytes are received from a master transmitter. To initialize the slave receiver mode, user should write the Slave Address Register (I2ADR) and write the I^2C Control Set Register (I2CONSET) as shown in Figure 22.

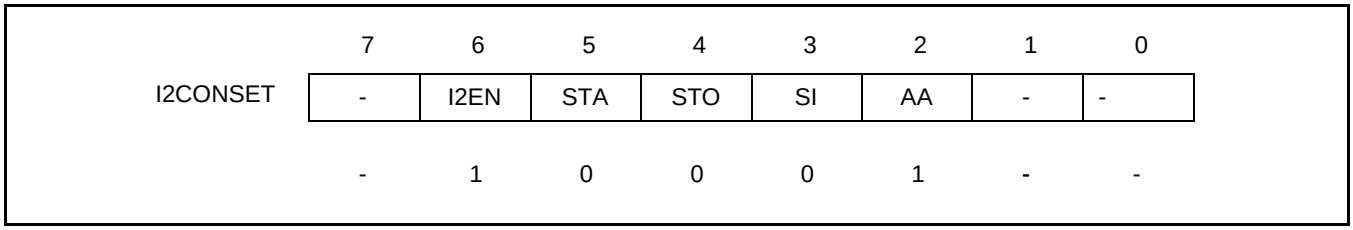


Figure 22: Slave Mode Configuration

I2EN must be set to 1 to enable the I^2C function. AA bit must be set to 1 to acknowledge its own slave address or the general call address. The STA, STO and SI bits are set to 0.

After I2ADR and I2CONSET are initialized, the I^2C interface waits until it is addressed by its own address or general address followed by the data direction bit. If the direction bit is 1(R), it enters slave transmitter mode. After the address and direction bit have been received, the SI bit is set and a valid status code can be read from the Status Register(I2STAT). Refer to Table 5 in "80C51 Family Derivatives 8XC552/562 Overview" datasheet available on-line at

http://www.semiconductors.philips.com/acrobat/various/8XC552_562OVERVIEW_2.pdf

for the status codes and actions.

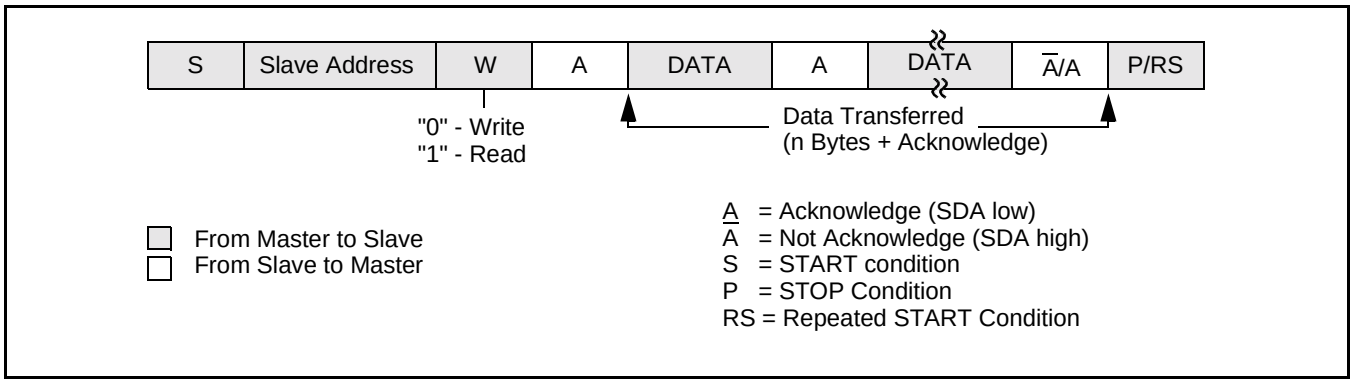


Figure 23: Format of slave receiver mode

Slave Transmitter Mode:

The first byte is received and handled as in the slave receiver mode. However, in this mode, the direction bit will indicate that the transfer direction is reversed. Serial data is transmitted via SDA while the serial clock is input through SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer. In a given application, I^2C may operate as a master and as a slave. In the slave mode, the I^2C hardware looks for its own slave address and the general call address. If one of these addresses is detected, an interrupt is requested. When the microcontroller wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave action is not interrupted. If bus arbitration is lost in the master mode, I^2C switches to the slave mode immediately and can detect its own slave address in the same serial transfer.

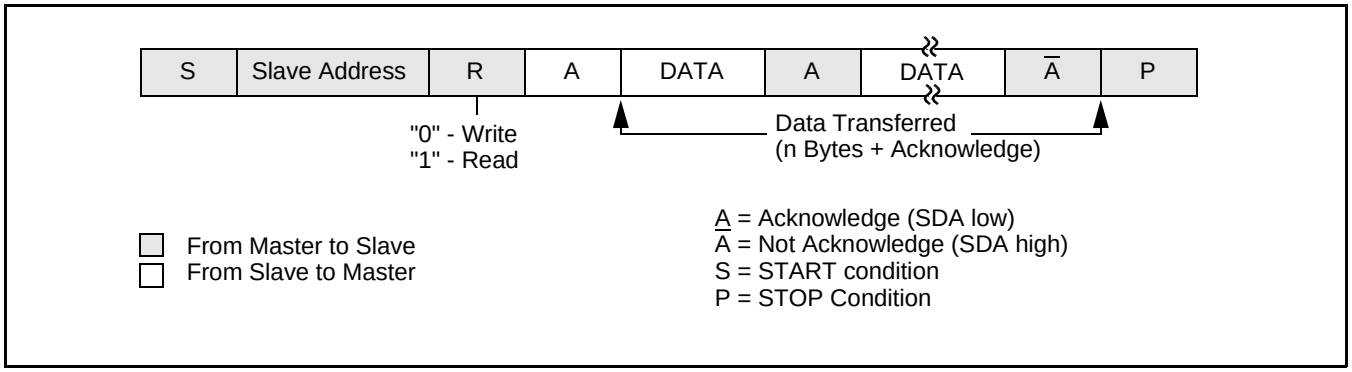


Figure 24: Format of slave transmitter mode

PIN DESCRIPTION

Table 85: I2C Pin Description

Pin Name	Type	Description
SDA	Input/ Output	**Serial Data.** I^2C data input and output. The associated port pin has an open drain output in order to conform to I^2C specifications.
SCL	Input/ Output	**Serial Clock.** I^2C clock input and output. The associated port pin has an open drain output in order to conform to I^2C specifications.

REGISTER DESCRIPTION

The I^2C interface contains 7 registers as shown in Table 86. below.

Table 86: I^2C Register Map

Address	Name	Description	Access	Reset Value*
0xE001C000	I2CONSET	I^2C Control Set Register	Read/Set	0
0xE001C004	I2STAT	I^2C Status Register	Read Only	0xF8
0xE001C008	I2DAT	I^2C Data Register	Read/Write	0
0xE001C00C	I2ADR	I^2C Slave Address Register	Read/Write	0
0xE001C010	I2SCLH	SCL Duty Cycle Register High Half Word	Read/Write	0x04
0xE001C014	I2SCLL	SCL Duty Cycle Register Low Half Word	Read/Write	0x04
0xE001C018	I2CONCLR	I^2C Control Clear Register	Clear Only	NA

*Reset Value refers to the data stored in used bits only. It does not include reserved bits content.

I²C Control Set Register (I2CONSET - 0xE001C000)

AA is the Assert Acknowledge Flag. When set to 1, an acknowledge (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line on the following situations:

1. The address in the Slave Address Register has been received.
2. The general call address has been received while the general call bit(GC) in I2ADR is set.
3. A data byte has been received while the I²C is in the master receiver mode.
4. A data byte has been received while the I²C is in the addressed slave receiver mode

The AA bit can be cleared by writing 1 to the AAC bit in the I2CONCLR register. When AA is 0, a not acknowledge (high level to SDA) will be returned during the acknowledge clock pulse on the SCL line on the following situations:

1. A data byte has been received while the I²C is in the master receiver mode.
2. A data byte has been received while the I²C is in the addressed slave receiver mode.

SI is the I²C Interrupt Flag. This bit is set when one of the 25 possible I²C states is entered. Typically, the I²C interrupt should only be used to indicate a start condition at an idle slave device, or a stop condition at an idle master device (if it is waiting to use the I²C bus). SI is cleared by writing a 1 to the SIC bit in I2CONCLR register.

STO is the STOP flag. Setting this bit causes the I²C interface to transmit a STOP condition in master mode, or recover from an error condition in slave mode. When STO is 1 in master mode, a STOP condition is transmitted on the I²C bus. When the bus detects the STOP condition, STO is cleared automatically.

In slave mode, setting this bit can recover from an error condition. In this case, no STOP condition is transmitted to the bus. The hardware behaves as if a STOP condition has been received and it switches to "not addressed" slave receiver mode. The STO flag is cleared by hardware automatically.

STA is the START flag. Setting this bit causes the I²C interface to enter master mode and transmit a START condition or transmit a repeated START condition if it is already in master mode.

When STA is 1and the I²C interface is not already in master mode, it enters master mode, checks the bus and generates a START condition if the bus is free. If the bus is not free, it waits for a STOP condition (which will free the bus) and generates a START condition after a delay of a half clock period of the internal clock generator. If the I²C interface is already in master mode and data has been transmitted or received, it transmits a repeated START condition. STA may be set at any time, including when the I²C interface is in an addressed slave mode.

STA can be cleared by writing 1 to the STAC bit in the I2CONCLR register. When STA is 0, no START condition or repeated START condition will be generated.

If STA and STO are both set, then a STOP condition is transmitted on the I²C bus if it the interface is in master mode, and transmits a START condition thereafter. If the I²C interface is in slave mode, an internal STOP condition is generated, but is not transmitted on the bus.

I2EN I^2C Interface Enable. When I2EN is 1, the I^2C function is enabled. I2EN can be cleared by writing 1 to the I2ENC bit in the I2CONCLR register. When I2EN is 0, the I^2C function is disabled.

Table 87: I^2C Control Set Register (I2CONSET - 0xE001C000)

I2CONSET	Function	Description	Reset Value
0	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
1	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	AA	Assert acknowledge flag	0
3	SI	I^2C interrupt flag	0
4	STO	STOP flag	0
5	STA	START flag	0
6	I2EN	I^2C interface enable	0
7	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

I^2C Control Clear Register (I2CONCLR - 0xE001C018)

Table 88: I^2C Control Clear Register (I2CONCLR - 0xE001C018)

I2CONCLR	Function	Description	Reset Value
0	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
1	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	AAC	Assert Acknowledge Clear bit. Writing a 1 to this bit clears the AA bit in the I2CONSET register. Writing 0 has no effect.	NA
3	SIC	I^2C Interrupt Clear Bit. Writing a 1 to this bit clears the SI bit in the I2CONSET register. Writing 0 has no effect.	NA
4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
5	STAC	Start flag clear bit. Writing a 1 to this bit clears the STA bit in the I2CONSET register. Writing 0 has no effect.	NA
6	I2ENC	I^2C interface disable. Writing a 1 to this bit clears the I2EN bit in the I2CONSET register.Writing 0 has no effect.	NA
7	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

I²C Status Register (I2STAT - 0xE001C004)

This is a read-only register. It contains the status code of the I²C interface. The least three bits are always 0. There are 26 possible status codes. When the code is F8H, there is no relevant information available and the SI bit is not set. All other 25 status codes correspond to defined I²C states. When any of these states entered, SI bit will be set. Refer to Table 3 to Table 6 in "80C51 Family Derivatives 8XC552/562 Overview" datasheet available on-line at

http://www.semiconductors.philips.com/acrobat/various/8XC552_562OVERVIEW_2.pdf

for a complete list of status codes.

Table 89: I²C Status Register (I2STAT - 0xE001C004)

I2STAT	Function	Description	Reset Value
2:0	Status	These bits are always 0	0
7:3	Status	Status bits	1

I²C Data Register (I2DAT - 0xE001C008)

This register contains the data to be transmitted or the data just received. The CPU can read and write to this register while it is not in the process of shifting a byte. This register can be accessed only when SI bit is set. Data in I2DAT remains stable as long as the SI bit is set. Data in I2DAT is always shifted from right to left: the first bit to be transmitted is the MSB (bit 7), and after a byte has been received, the first bit of received data is located at the MSB of I2DAT.

Table 90: I²C Data Register (I2DAT - 0xE001C008)

I2DAT	Function	Description	Reset Value
7:0	Data	Transmit/Receive data bits	0

I²C Slave Address Register (I2ADR - 0xE001C00C)

This register is readable and writable, and is only used when the I²C is set to slave mode. In master mode, this register has no effect. The LSB of I2ADR is the general call bit. When this bit is set, the general call address (00h) is recognized.

Table 91: I²C Slave Address Register (I2ADR - 0xE001C00C)

I2ADR	Function	Description	Reset Value
0	GC	General Call bit	0
7:1	Address	Slave mode address	0

I^2C SCL Duty Cycle Registers (I2SCLH - 0xE001C010 and I2SCLL - 0xE001C014)

Software must set values for registers I2SCLH and I2SCLL to select the appropriate data rate. I2SCLH defines the number of pclk cycles for SCL high, I2SCLL defines the number of pclk cycles for SCL low. The frequency is determined by the following formula:

Bit Frequency = f_{CLK} / (I2SCLH + I2SCLL)

Where f_{CLK} is the frequency of pclk.

The values for I2SCLL and I2SCLH don't have to be the same. Software can set different duty cycles on SCL by setting these two registers. But the value of the register must ensure that the data rate is in the I^2C data rate range of 0 through 400KHz. So the value of I2SCLL and I2SCLH has some restrictions. Each register value should be greater than or equal to 4.

Table 92: I^2C SCL High Duty Cycle Register (I2SCLH - 0xE001C010)

I2SCLH	Function	Description	Reset Value
15:0	Count	Count for SCL HIGH time period selection	0x 0004

Table 93: I^2C SCL Low Duty Cycle Register (I2SCLL - 0xE001C014)

I2SCLL	Function	Description	Reset Value
15:0	Count	Count for SCL LOW time period selection	0x 0004

Table 94: I2C Clock Rate Selections for VPB Clock Divider = 1

I2SCLL+ I2SCLH	Bit Frequency (kHz) At f_{CCLK} (MHz) & VPB Clock Divider = 1			
	16	20	40	60
8	-	-	-	-
10	-	-	-	-
25	-	-	-	-
50	320.0	400.0	-	-
75	213.333	266.667	-	-
100	160.0	200.0	400.0	-
160	100.0	125.0	250.0	375.0
200	80.0	100.0	200.0	300.0
320	50.0	62.5	125.0	187.5
400	40.0	50.0	100.0	150.0
510	31.373	39.216	78.431	117.647
800	20.0	25.0	50.0	75.0
1280	12.5	15.625	31.25	46.875

Table 95: I2C Clock Rate Selections for VPB Clock Divider = 2

I2SCLL+ I2SCLH	Bit Frequency (kHz) At f_{CCLK} (MHz) & VPB Clock Divider = 2			
	16	20	40	60
8	-	-	-	-
10	-	-	-	-
25	320.0	400.0	-	-
50	160.0	200.0	400.0	-
75	106.667	133.333	266.667	400.0
100	80.0	100.0	200.0	300.0
160	50.0	62.5	125.0	187.5
200	40.0	50.0	100.0	150.0
320	25.0	31.25	62.5	93.75
400	20.0	25.0	50.0	75.0
510	15.686	19.608	39.216	58.824
800	10.0	12.5	25.0	37.5
1280	6.25	7.813	15.625	23.438

Table 96: I2C Clock Rate Selections for VPB Clock Divider = 4

I2SCLL+ I2SCLH	Bit Frequency (kHz) At f_{CCLK} (MHz) & VPB Clock Divider = 4			
	16	20	40	60
8	500.0	-	-	-
10	400.0	-	-	-
25	160.0	200.0	400.0	-
50	80.0	100.0	200.0	300.0
75	53.333	66.667	133.333	200.0
100	40.0	50.0	100.0	150.0
160	25.0	31.25	62.5	93.75
200	20.0	25.0	50.0	75.0
320	12.5	15.625	31.25	46.875
400	10.0	12.5	25.0	37.5
510	7.843	9.804	19.608	29.412
800	5.0	6.25	12.5	18.75
1280	3.125	3.906	7.813	11.719

ARCHITECTURE

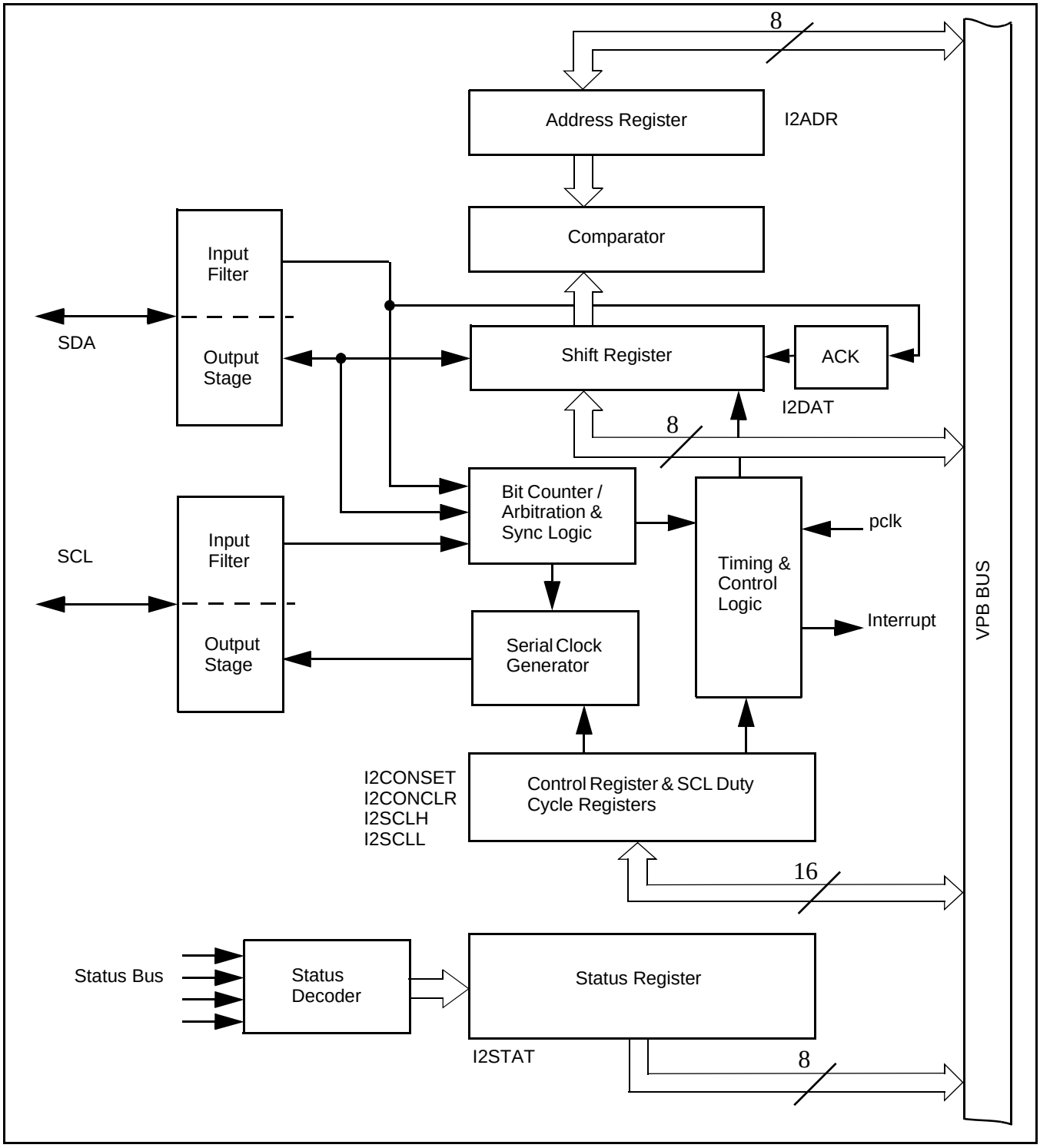


Figure 25: I²C Architecture

12. SPI INTERFACE

FEATURES

- Compliant with Serial Peripheral Interface (SPI) specification.
- Synchronous, Serial, Full Duplex Communication.
- Combined SPI master and slave.
- Maximum data bit rate of one eighth of the input clock rate.

DESCRIPTION

SPI Overview

The SPI is a full duplex serial interface. It is designed to be able to handle multiple masters and slaves being connected to a given bus. Only a single master and a single slave can communicate on the interface during a given data transfer. During a data transfer the master always sends a byte of data to the slave, and the slave always sends a byte of data to the master.

SPI Data Transfers

Figure 26 is a timing diagram that illustrates the four different data transfer formats that are available with the SPI. This timing diagram illustrates a single 8 bit data transfer. The first thing one should notice in this timing diagram is that it is divided into three horizontal parts. The first part describes the SCK and SSEL signals. The second part describes the MOSI and MISO signals when the CPHA variable is 0. The third part describes the MOSI and MISO signals when the CPHA variable is 1.

In the first part of the timing diagram, note two points. First, the SPI is illustrated wit CPOL set to both 0 and 1. The second point to note is the activation and de-activation of the SSEL signal. When CPHA = 1, the SSEL signal will always go inactive between data transfers. This is not guaranteed when CPHA = 0 (the signal can remain active).

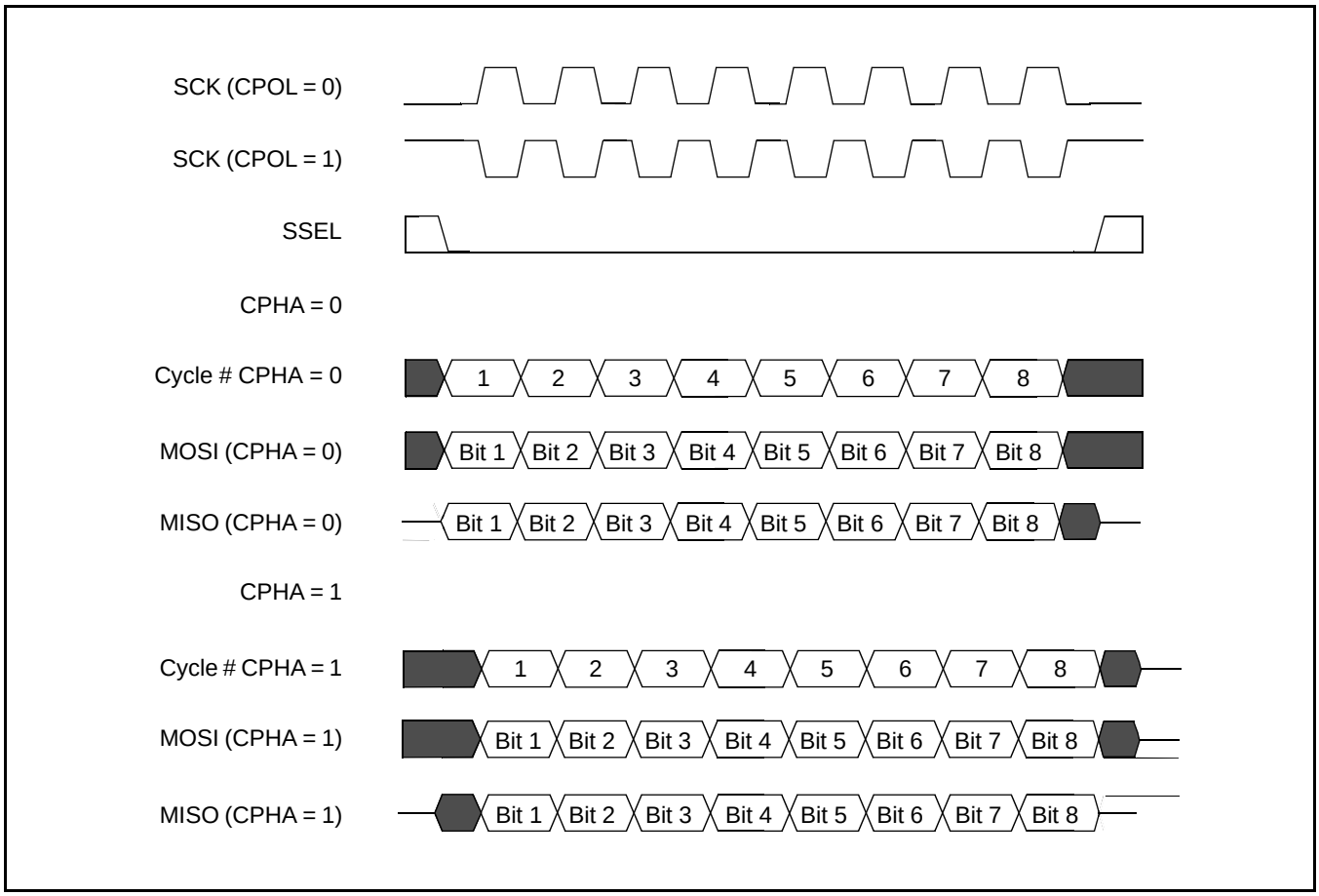


Figure 26: SPI Data Transfer Format (CPHA = 0 and CPHA = 1)

The data and clock phase relationships are summarized in Table 97. This table summarizes the following for each setting of CPOL and CPHA.

- When the first data bit is driven.
- When all other data bits are driven.
- When data is sampled.

Table 97: SPI Data To Clock Phase Relationship

CPOL And CPHA Settings	First Data Driven	Other Data Driven	Data Sampled
CPOL = 0, CPHA = 0	Prior to first SCK rising edge	SCK falling edge	SCK rising edge
CPOL = 0, CPHA = 1	First SCK rising edge	SCK rising edge	SCK falling edge
CPOL = 1, CPHA = 0	Prior to first SCK falling edge	SCK rising edge	SCK falling edge
CPOL = 1, CPHA = 1	First SCK falling edge	SCK falling edge	SCK rising edge

The definition of when an 8 bit transfer starts and stops is dependent on whether a device is a master or a slave, and the setting of the CPHA variable.

When a device is a master, the start of a transfer is indicated by the master having a byte of data that is ready to be transmitted. At this point, the master can activate the clock, and begin the transfer. The transfer ends when the last clock cycle of the transfer is complete.

When a device is a slave, and CPHA is set to 0, the transfer starts when the SSEL signal goes active, and ends when SSEL goes inactive. When a device is a slave, and CPHA is set to 1, the transfer starts on the first clock edge when the slave is selected, and ends on the last clock edge where data is sampled.

SPI Peripheral Details

General Information

There are four registers that control the SPI peripheral. They are described in detail in “Register Description” section.

The SPI control register contains a number of programmable bits used to control the function of the SPI block. The settings for this register must be set up prior to a given data transfer taking place.

The SPI status register contains read only bits that are used to monitor the status of the SPI interface, including normal functions, and exception conditions. The primary purpose of this register is to detect completion of a data transfer. This is indicated by the SPIF bit. The remaining bits in the register are exception condition indicators. These exceptions will be described later in this section.

The SPI data register is used to provide the transmit and receive data bytes. An internal shift register in the SPI block logic is used for the actual transmission and reception of the serial data. Data is written to the SPI data register for the transmit case. There is no buffer between the data register and the internal shift register. A write to the data register goes directly into the internal shift register. Therefore, data should only be written to this register when a transmit is not currently in progress. Read data is buffered. When a transfer is complete, the receive data is transferred to a single byte data buffer, where it is later read. A read of the SPI data register returns the value of the read data buffer.

The SPI clock counter register controls the clock rate when the SPI block is in master mode. This needs to be set prior to a transfer taking place, when the SPI block is a master. This register has no function when the SPI block is a slave.

The I/Os for this implementation of SPI are standard CMOS I/Os. The open drain SPI option is not implemented in this design. When a device is set up to be a slave, its I/Os are only active when it is selected by the SSEL signal being active.

Master Operation

The following sequence describes how one should process a data transfer with the SPI block when it is set up to be the master. This process assumes that any prior data transfer has already completed.

1. Set the SPI clock counter register to the desired clock rate.
2. Set the SPI control register to the desired settings.
3. Write the data to transmitted to the SPI data register. This write starts the SPI data transfer.
4. Wait for the SPIF bit in the SPI status register to be set to 1. The SPIF bit will be set after the last cycle of the SPI data transfer.
5. Read the SPI status register.
6. Read the received data from the SPI data register (optional).
7. Go to step 3 if more data is required to transmit.

Note that a read or write of the SPI data register is required in order to clear the SPIF status bit. Therefore, if the optional read of the SPI data register does not take place, a write to this register is required in order to clear the SPIF status bit.

Slave Operation

The following sequence describes how one should process a data transfer with the SPI block when it is set up to be a slave. This process assumes that any prior data transfer has already completed. It is required that the system clock driving the SPI logic be at least 8X faster than the SPI.

1. Set the SPI control register to the desired settings.
2. Write the data to transmitted to the SPI data register (optional). Note that this can only be done when a slave SPI transfer is not in progress.
3. Wait for the SPIF bit in the SPI status register to be set to 1. The SPIF bit will be set after the last sampling clock edge of the SPI data transfer.
4. Read the SPI status register.
5. Read the received data from the SPI data register (optional).
6. Go to step 2 if more data is required to transmit.

Note that a read or write of the SPI data register is required in order to clear the SPIF status bit. Therefore, at least one of the optional reads or writes of the SPI data register must take place, in order to clear the SPIF status bit.

Exception Conditions

Read Overrun - A read overrun occurs when the SPI block internal read buffer contains data that has not been read by the processor, and a new transfer has completed. The read buffer containing valid data is indicated by the SPIF bit in the status register being active. When a transfer completes, the SPI block needs to move the received data to the read buffer. If the SPIF bit is active (the read buffer is full), the new receive data will be lost, and the read overrun (ROVR) bit in the status register will be activated.

Write Collision - As stated previously, there is no write buffer between the SPI block bus interface, and the internal shift register. As a result, data must not be written to the SPI data register when a SPI data transfer is currently in progress. The time frame where data cannot be written to the SPI data register is from when the transfer starts, until after the status register has been read when the SPIF status is active. If the SPI data register is written in this time frame, the write data will be lost, and the write collision (WCOL) bit in the status register will be activated.

Mode Fault - The SSEL signal must always be inactive when the SPI block is a master. If the SSEL signal goes active, when the SPI block is a master, this indicates another master has selected the device to be a slave. This condition is known as a mode fault. When a mode fault is detected, the mode fault (MODF) bit in the status register will be activated, the SPI signal drivers will be de-activated, and the SPI mode will be changed to be a slave.

Slave Abort - A slave transfer is considered to be aborted, if the SSEL signal goes inactive before the transfer is complete. In the event of a slave abort, the transmit and receive data for the transfer that was in progress are lost, and the slave abort (ABRT) bit in the status register will be activated.

PIN DESCRIPTION

Table 98: SPI Pin Description

Pin Name	Type	Pin Description
SCK	Input/ Output	**Serial Clock.** The SPI is a clock signal used to synchronize the transfer of data across the SPI interface. The SPI is always driven by the master and received by the slave. The clock is programmable to be active high or active low. The SPI is only active during a data transfer. Any other time, it is either in its inactive state, or tri-stated.
SSEL	Input	**Slave Select.** The SPI slave select signal is an active low signal that indicates which slave is currently selected to participate in a data transfer. Each slave has its own unique slave select signal input. The SSEL must be low before data transactions begin and normally stays low for the duration of the transaction. If the SSEL signal goes high any time during a data transfer, the transfer is considered to be aborted. In this event, the slave returns to idle, and any data that was received is thrown away. There are no other indications of this exception. This signal is not directly driven by the master. It could be driven by a simple general purpose I/O under software control.
MISO	Input/ Output	**Master In Slave Out.** The MISO signal is a unidirectional signal used to transfer serial data from the slave to the master. When a device is a slave, serial data is output on this signal. When a device is a master, serial data is input on this signal. When a slave device is not selected, the slave drives the signal high impedance.
MOSI	Input/ Output	**Master Out Slave In.** The MOSI signal is a unidirectional signal used to transfer serial data from the master to the slave. When a device is a master, serial data is output on this signal. When a device is a slave, serial data is input on this signal.

REGISTER DESCRIPTION

The SPI contains 5 registers as shown in Table 99. All registers are byte, half word and word accessible.

Table 99: SPI Register Map

Address	Name	Description	Access	Reset Value*
0xE0020000	SPCR	SPI Control Register. This register controls the operation of the SPI.	Read/Write	0
0xE0020004	SPSR	SPI Status Register. This register shows the status of the SPI.	Read Only	0
0xE0020008	SPDR	SPI Data Register. This bi-directional register provides the transmit and receive data for the SPI. Transmit data is provided to the SPI by writing to this register. Data received by the SPI can be read from this register.	Read/Write	0
0xE002000C	SPCCR	SPI Clock Counter Register. This register controls the frequency of a master's SCK.	Read/Write	0
0xE002001C	SPINT	SPI Interrupt Flag. This register contains the interrupt flag for the SPI interface.	Read/Write	0

*Reset Value refers to the data stored in used bits only. It does not include reserved bits content.

SPI Control Register (SPCR - 0xE0020000)

The SPCR register controls the operation of the SPI as per the configuration bits setting.

Table 100: SPI Control Register (SPCR - 0xE0020000)

SPCR	Function	Description	Reset Value
2:0	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	CPHA	Clock phase control determines the relationship between the data and the clock on SPI transfers, and controls when a slave transfer is defined as starting and ending. When 1, data is sampled on the second clock edge of the SCK. A transfer starts with the first clock edge, and ends with the last sampling edge when the SSEL signal is active. When 0, data is sampled on the first clock edge of SCK. A transfer starts and ends with activation and deactivation of the SSEL signal.	0
4	CPOL	Clock polarity control. When 1, SCK is active low. When 0, SCK is active high.	0
5	MSTR	Master mode select. When 1, the SPI operates in Master mode. When 0, the SPI operates in Slave mode.	0
6	LSBF	LSB First controls which direction each byte is shifted when transferred. When 1, SPI data is transferred LSB (bit 0) first. When 0, SPI data is transferred MSB (bit 7) first.	0
7	SPIE	Serial peripheral interrupt enable. When 1, a hardware interrupt is generated each time the SPIF or MODF bits are activated. When 0, SPI interrupts are inhibited.	0

SPI Status Register (SPSR - 0xE0020004)

The SPSR register controls the operation of the SPI as per the configuration bits setting.

Table 101: SPI Status Register (SPSR - 0xE0020004)

SPSR	Function	Description	Reset Value
2:0	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	ABRT	Slave abort. When 1, this bit indicates that a slave abort has occurred. This bit is cleared by reading this register.	0
4	MODF	Mode fault. when 1, this bit indicates that a Mode fault error has occurred. This bit is cleared by reading this register, then writing the SPI control register.	0
5	ROVR	Read overrun. When 1, this bit indicates that a read overrun has occurred. This bit is cleared by reading this register.	0
6	WCOL	Write collision. When 1, this bit indicates that a write collision has occurred. This bit is cleared by reading this register, then accessing the SPI data register.	0
7	SPIF	SPI transfer complete flag. When 1, this bit indicates when a SPI data transfer is complete. When a master, this bit is set at the end of the last cycle of the transfer. When a slave, this bit is set on the last data sampling edge of the SCK. This bit is cleared by first reading this register, then accessing the SPI data register. **Note:** this is not the SPI interrupt flag. This flag is found in the SPINT registrer.	0

SPI Data Register (SPDR - 0xE0020008)

This bi-directional data register provides the transmit and receive data for the SPI. Transmit data is provided to the SPI by writing to this register. Data received by the SPI can be read from this register. When a master, a write to this register will start a SPI data transfer. Writes to this register will be blocked from when a data transfer starts to when the SPIF status bit is set, and the status register has not been read.

Table 102: SPI Data Register (SPDR - 0xE0020008)

SPDR	Function	Description	Reset Value
7:0	Data	SPI Bi-directional data port	0

SPI Clock Counter Register (SPCCR - 0xE002000C)

This register controls the frequency of a master's SCK. The register indicates the number of pclk cycles that make up an SPI clock. The value of this register must always be an even number. As a result, bit 0 must always be 0. The value of the register must also always be greater than or equal to 8. Violations of this can result in unpredictable behavior.

Table 103: SPI Clock Counter Register (SPCCR - 0xE002000C)

SPCCR	Function	Description	Reset Value
7:0	Counter	SPI Clock counter setting	0

The SPI rate may be calculated as: PCLK rate / SPCCR value. The pclk rate is CCLK / VPB divider rate as determined by the VPBDIV register contents.

SPI Interrupt Register (SPINT - 0xE002001C)

This register contains the interrupt flag for the SPI interface.

Table 104: SPI Interrupt Register (SPINT - 0xE002001C)

SPINT	Function	Description	Reset Value
0	SPI Interrupt	SPI interrupt flag. Set by the SPI interface to generate an interrupt. Cleared by writing a 1 to this bit.	0
7:1	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

ARCHITECTURE

The block diagram of the SPI is shown in the Figure 27.

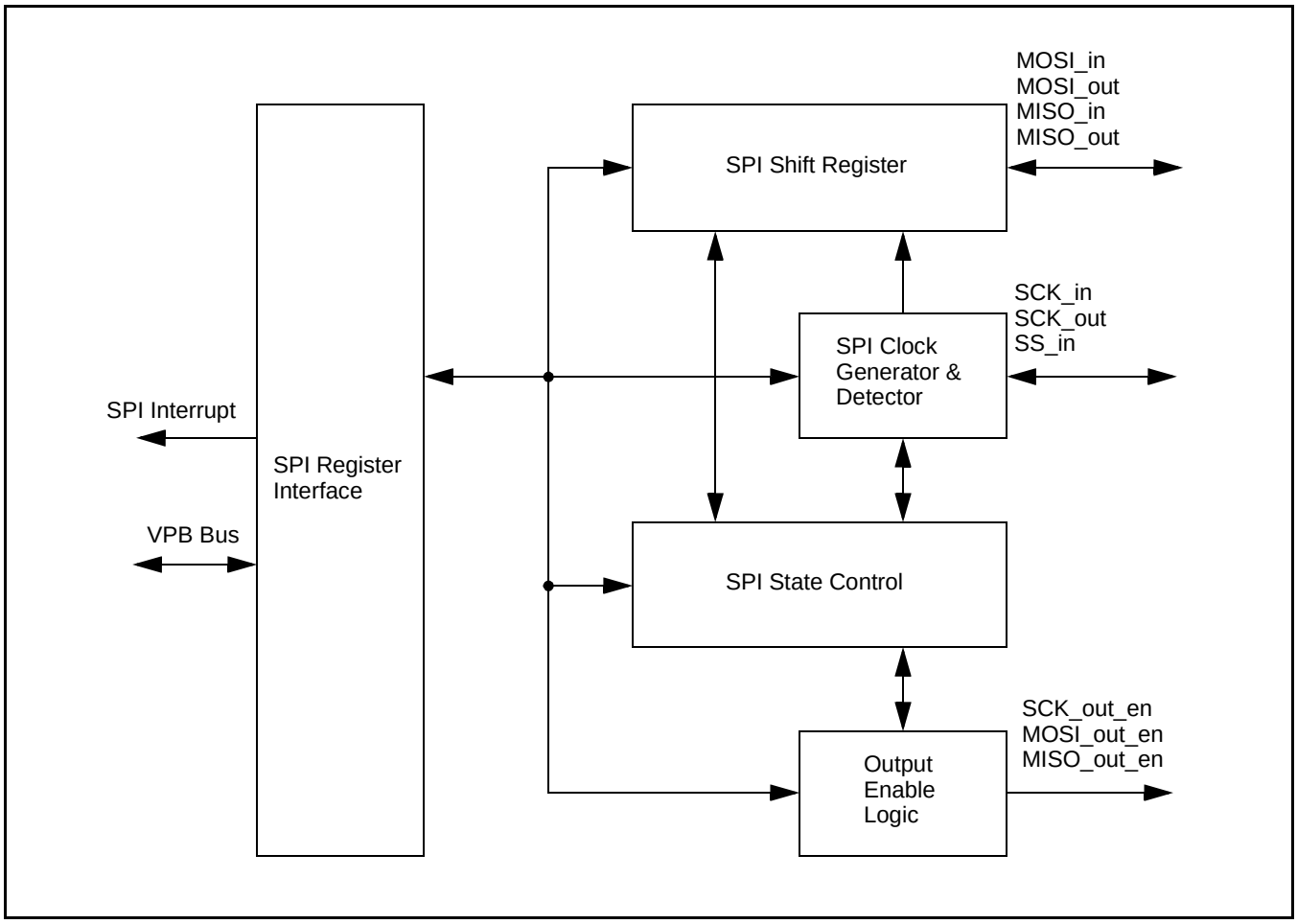


Figure 27: SPI Block Diagram

13. TIMER 0 AND TIMER 1

Timer 0 and Timer 1 are functionally identical except for the peripheral base address.

FEATURES

- A 32-bit Timer/Counter with a programmable 32-bit Prescaler.
- Up to four (Timer 1) and three (Timer 0) 32-bit capture channels, that can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt.
- Four 32-bit match registers that allow:
  - Continuous operation with optional interrupt generation on match.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.
- Up to four (Timer 1) and three (Timer 0) external outputs corresponding to match registers, with the following capabilities:
  - Set low on match.
  - Set high on match.
  - Toggle on match.
  - Do nothing on match.

APPLICATIONS

- Interval Timer for counting internal events.
- Pulse Width Demodulator via Capture inputs.
- Free running timer.

DESCRIPTION

The Timer is designed to count cycles of the peripheral clock (pclk) and optionally generate interrupts or perform other actions at specified timer values, based on four match registers. It also includes four capture inputs to trap the timer value when an input signal transitions, optionally generating an interrupt.

Due to the limited number of pins on the LPC2106/2105/2104, only three of the Capture inputs and Match outputs of Timer 0 are connected to device pins.

PIN DESCRIPTION

Table 105 gives a brief summary of each of the Timer related pins.

Table 105: Pin summary

Pin name	Pin direction	Pin Description
CAP0.2..0 CAP1.3..0	Input	**Capture Signals**- A transition on a capture pin can be configured to load one of the Capture Registers with the value in the Timer Counter and optionally generate an interrupt.
MAT0.0 MAT1.0	Output	**External Match Output 0/1-** When match register 0/1 (MR0/1) equals the timer counter (TC) this output can either toggle, go low, go high, or do nothing. The External Match Register (EMR) controls the functionality of this output.
MAT0.1 MAT1.1	Output	**External Match Output 1-** See the MAT0/MAT1 description above.
MAT0.2 MAT1.2	Output	**External Match Output 2-** See the MAT0/MAT1 description above.
MAT1.3	Output	**External Match Output 3-** See the MAT1 description above.

REGISTER DESCRIPTION

Each Timer contains the registers shown in Table 106. More detailed descriptions follow.

Table 106: Timer 0 and Timer 1 Register Map

Generic Name	Timer 0 Address & Name	Timer 1 Address & Name	Description	Access	Reset Value*
IR	0xE0004000 T0IR	0xE0008000 T1IR	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight (Timer 1) or seven (Timer 0) possible interrupt sources are pending.	R/W	0
TCR	0xE0004004 T0TCR	0xE0008004 T1TCR	Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	R/W	0
TC	0xE0004008 T0TC	0xE0008008 T1TC	Timer Counter. The 32-bit TC is incremented every PR+1 cycles of pclk. The TC is controlled through the TCR.	RW	0
PR	0xE000400C T0PR	0xE000800C T1PR	Prescale Register. The TC is incremented every PR+1 cycles of pclk.	R/W	0
PC	0xE0004010 T0PC	0xE0008010 T1PC	Prescale Counter. The 32-bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented.	R/W	0
MCR	0xE0004014 T0MCR	0xE0008014 T1MCR	Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	R/W	0
MR0	0xE0004018 T0MR0	0xE0008018 T1MR0	Match Register 0. MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC.	R/W	0
MR1	0xE000401C T0MR1	0xE000801C T1MR1	Match Register 1. See MR0 description.	R/W	0
MR2	0xE0004020 T0MR2	0xE0008020 T1MR2	Match Register 2. See MR0 description.	R/W	0
MR3	0xE0004024 T0MR3	0xE0008024 T1MR3	Match Register 3. See MR0 description.	R/W	0
CCR	0xE0004028 T0CCR	0xE0008028 T1CCR	Capture Control Register. The CCR controls which edges of the capture inputs are used to load the Capture Registers and whether or not an interrupt is generated when a capture takes place.	R/W	0
CR0	0xE000402C T0CR0	0xE000802C T1CR0	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the capture[0] signal.	RO	0
CR1	0xE0004030 T0CR1	0xE0008030 T1CR1	Capture Register 1. See CR0 description.	RO	0
CR2	0xE0004034 T0CR2	0xE0008034 T1CR2	Capture Register 2. See CR0 description.	RO	0
CR3	0xE0004038 T0CR3	0xE0008038 T1CR3	Capture Register 3. See CR0 description. Not usable on Timer 0.	RO	0
EMR	0xE000403C T0EMR	0xE000803C T1EMR	External Match Register. The EMR controls the external match pins MATn.	R/W	0

*Reset Value refers to the data stored in used bits only. It does not include reserved bits content.

Interrupt Register (IR: Timer 0 - T0IR: 0xE0004000; Timer 1 - T1IR: 0xE0008000)

The Interrupt Register consists of four bits for the match interrupts and four bits (three for Timer 0) for the capture interrupts. If an interrupt is generated then the corresponding bit in the IR will be high. Otherwise, the bit will be low. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect.

Table 107: Interrupt Register (IR: Timer 0 - T0IR: 0xE0004000; Timer 1 - T1IR: 0xE0008000)

IR	Function	Description	Reset Value
0	MR0 Interrupt	Interrupt flag for match channel 0.	0
1	MR1 Interrupt	Interrupt flag for match channel 1.	0
2	MR2 Interrupt	Interrupt flag for match channel 2.	0
3	MR3 Interrupt	Interrupt flag for match channel 3.	0
4	CR0 Interrupt	Interrupt flag for capture channel 0 event.	0
5	CR1 Interrupt	Interrupt flag for capture channel 1 event.	0
6	CR2 Interrupt	Interrupt flag for capture channel 2 event.	0
7	CR3 Interrupt	Interrupt flag for capture channel 3 event.	0

Timer Control Register (TCR: Timer 0 - T0TCR: 0xE0004004; Timer 1 - T1TCR: 0xE0008004)

The Timer Control Register (TCR) is used to control the operation of the Timer Counter.

Table 108: Timer Control Register (TCR: Timer 0 - T0TCR: 0xE0004004; Timer 1 - T1TCR: 0xE0008004)

TCR	Function	Description	Reset Value
0	Counter Enable	When one, the Timer Counter and Prescale Counter are enabled for counting. When zero, the counters are disabled.	0
1	Counter Reset	When one, the Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of pclk. The counters remain reset until TCR[1] is returned to zero.	0

Timer Counter (TC: Timer 0 - T0TC: 0xE0004008; Timer 1 - T1TC: 0xE0008008)

The 32-bit Timer Counter is incremented when the Prescale Counter reaches its terminal count. Unless it is reset before reaching its upper limit, the TC will count up through the value 0xFFFFFFFF and then wrap back to the value 0x00000000. This event does not cause an interrupt, but a Match register can be used to detect an overflow if needed.

Prescale Register (PR: Timer 0 - T0PC: 0xE000400C; Timer 1 - T1PC: 0xE000800C)

The 32-bit Prescale Register specifies the maximum value for the Prescale Counter.

Prescale Counter Register (PC: Timer 0 - T0PC: 0xE0004010; Timer 1 - T1PC: 0xE0008010)

The 32-bit Prescale Counter controls division of pclk by some constant value before it is applied to the Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The Prescale Counter is incremented on every pclk. When it reaches the value stored in the Prescale Register, the Timer Counter is incremented and the Prescale Counter is reset on the next pclk. This causes the TC to increment on every pclk when PR = 0, every 2 pclks when PR = 1, etc.

Match Registers (MR0 - MR3)

The Match register values are continuously compared to the Timer Counter value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the Timer Counter, or stop the timer. Actions are controlled by the settings in the MCR register.

Match Control Register (MCR: Timer 0 - T0MCR: 0xE0004014; Timer 1 - T1MCR: 0xE0008014)

The Match Control Register is used to control what operations are performed when one of the Match Registers matches the Timer Counter. The function of each of the bits is shown in Table 109.

Table 109: Match Control Register (MCR: Timer 0 - T0MCR: 0xE0004014; Timer 1 - T1MCR: 0xE0008014)

MCR	Function	Description	Reset Value
0	Interrupt on MR0	When one, an interrupt is generated when MR0 matches the value in the TC. When zero this interrupt is disabled.	0
1	Reset on MR0	When one, the TC will be reset if MR0 matches it. When zero this feature is disabled.	0
2	Stop on MR0	When one, the TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC. When zero this feature is disabled.	0
3	Interrupt on MR1	When one, an interrupt is generated when MR1 matches the value in the TC. When zero this interrupt is disabled.	0
4	Reset on MR1	When one, the TC will be reset if MR1 matches it. When zero this feature is disabled.	0
5	Stop on MR1	When one, the TC and PC will be stopped and TCR[0] will be set to 0 if MR1 matches the TC. When zero this feature is disabled	0
6	Interrupt on MR2	When one, an interrupt is generated when MR2 matches the value in the TC. When zero this interrupt is disabled.	0
7	Reset on MR2	When one, the TC will be reset if MR2 matches it. When zero this feature is disabled.	0
8	Stop on MR2	When one, the TC and PC will be stopped and TCR[0] will be set to 0 if MR2 matches the TC. When zero this feature is disabled	0
9	Interrupt on MR3	When one, an interrupt is generated when MR3 matches the value in the TC. When zero this interrupt is disabled	0
10	Reset on MR3	When one, the TC will be reset if MR3 matches it. When zero this feature is disabled.	0
11	Stop on MR3	When one, the TC and PC will be stopped and TCR[0] will be set to 0 if MR3 matches the TC. When zero this feature is disabled	0

Capture Registers (CR0 - CR3)

Each Capture register is associated with a device pin and may be loaded with the Timer Counter value when a specified event occurs on that pin. The settings in the Capture Control Register register determine whether the capture function is enabled, and whether a capture event happens on the rising edge of the associated pin, the falling edge, or on both edges.

Capture Control Register (CCR: Timer 0 - T0CCR: 0xE0004028; Timer 1 - T1CCR: 0xE0008028)

The Capture Control Register is used to control whether one of the four Capture Registers is loaded with the value in the Timer Counter when the capture event occurs, and whether an interrupt is generated by the capture event. Setting both the rising and falling bits at the same time is a valid configuration, resulting in a capture event for both edges.

Table 110: Capture Control Register (CCR: Timer 0 - T0CCR: 0xE0004028; Timer 1 - T1CCR: 0xE0008028)

CCR	Function	Description	Reset Value
0	Capture on capture[0] rising edge	When one, a sequence of 0 then 1 on capture[0] will cause CR0 to be loaded with the contents of the TC. When zero this feature is disabled.	0
1	Capture on capture[0] falling edge	When one, a sequence of 1 then 0 on capture[0] will cause CR0 to be loaded with the contents of TC. When zero this feature is disabled.	0
2	Interrupt on capture[0] event	When one, a CR0 load due to a capture[0] event will generate an interrupt. When zero this feature is disabled.	0
3	Capture on capture[1] rising edge	When one, a sequence of 0 then 1 on capture[1] will cause CR1 to be loaded with the contents of the TC. When zero this feature is disabled.	0
4	Capture on capture[1] falling edge	When one, a sequence of 1 then 0 on capture[1] will cause CR1 to be loaded with the contents of TC. When zero this feature is disabled.	0
5	Interrupt on capture[1] event	When one, a CR1 load due to a capture[1] event will generate an interrupt. When zero this feature is disabled.	0
6	Capture on capture[2] rising edge	When one, a sequence of 0 then 1 on capture[2] will cause CR2 to be loaded with the contents of the TC. When zero this feature is disabled.	0
7	Capture on capture[2] falling edge	When one, a sequence of 1 then 0 on capture[2] will cause CR2 to be loaded with the contents of TC. When zero this feature is disabled.	0
8	Interrupt on capture[2] event	When one, a CR2 load due to a capture[2] event will generate an interrupt. When zero this feature is disabled.	0
9	Capture on capture[3 rising edge	(Timer 1 only.) When one, a sequence of 0 then 1 on capture[3] will cause CR3 to be loaded with the contents of TC. When zero this feature is disabled.	0
10	Capture on capture[3] falling edge	(Timer 1 only.) When one, a sequence of 1 then 0 on capture[3] will cause CR3 to be loaded with the contents of TC. When zero this feature is disabled.	0
11	Interrupt on capture[3] event	(Timer 1 only.) When one, a CR3 load due to a capture[3] event will generate an interrupt. When zero this feature is disabled.	0

External Match Register (EMR: Timer 0 - T0EMR: 0xE000403C; Timer 1 - T1EMR: 0xE000803C)

The External Match Register provides both control and status of the external match pins M(0-3).

Table 111: External Match Register (EMR: Timer 0 - T0EMR: 0xE000403C; Timer 1 - T1EMR: 0xE000803C)

EMR	Function	Description	Reset Value
0	External Match 0	This bit reflects the state of output MAT0/1, whether or not this output is connected to its pin. When a match occurs for MR0, this output of the timer can either toggle, go low, go high, or do nothing. Bits EMR[4:5] control the functionality of this output.	0
1	External Match 1	This bit reflects the state of output MAT0/1, whether or not this output is connected to its pin. When a match occurs for MR1, this output of the timer can either toggle, go low, go high, or do nothing. Bits EMR[6:7] control the functionality of this output.	0
2	External Match 2	This bit reflects the state of output MAT0/1, whether or not this output is connected to its pin. When a match occurs for MR2, this output of the timer can either toggle, go low, go high, or do nothing. Bits EMR[8:9] control the functionality of this output.	0
3	External Match 3	This bit reflects the state of output MAT0/1. When a match occurs for MR3, this output of the timer can either toggle, go low, go high, or do nothing. Bits EMR[10:11] control the functionality of this output. **Note:** In the case of Timer 0, this output cannot be connected to a device pin.	0
5:4	External Match Control 0	Determines the functionality of External Match 0. Table 112 shows the encoding of these bits.	0
7:6	External Match Control 1	Determines the functionality of External Match 1. Table 112 shows the encoding of these bits.	0
9:8	External Match Control 2	Determines the functionality of External Match 2. Table 112 shows the encoding of these bits.	0
11:10	External Match Control 3	Determines the functionality of External Match 3. Table 112 shows the encoding of these bits.	0

Table 112: External Match Control

EMR[11:10], EMR[9:8], EMR[7:6], or EMR[5:4]	Function
0 0	Do Nothing
0 1	Clear corresponding External Match output to 0 (LOW if pinned out)
1 0	Set corresponding External Match output to 1(HIGH if pinned out)
1 1	Toggle corresponding External Match output

EXAMPLE TIMER OPERATION

Figure 28 shows a timer configured to reset the count and generate an interrupt on match. The prescaler is set to 2 and the match register set to 6. At the end of the timer cycle where the match occurs, the timer count is reset. This gives a full length cycle to the match value. The interrupt indicating that a match occurred is generated in the next clock after the timer reached the match value.

Figure 29 shows a timer configured to stop and generate an interrupt on match. The prescaler is again set to 2 and the match register set to 6. In the next clock after the timer reaches the match value, the timer enable bit in TCR is cleared, and the interrupt indicating that a match occurred is generated.

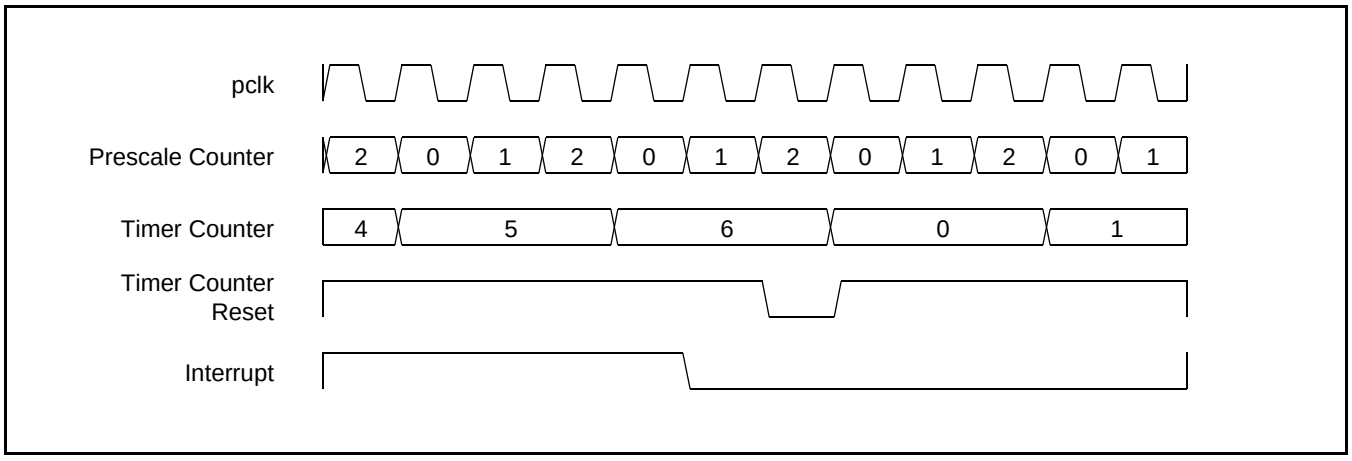


Figure 28: A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled.

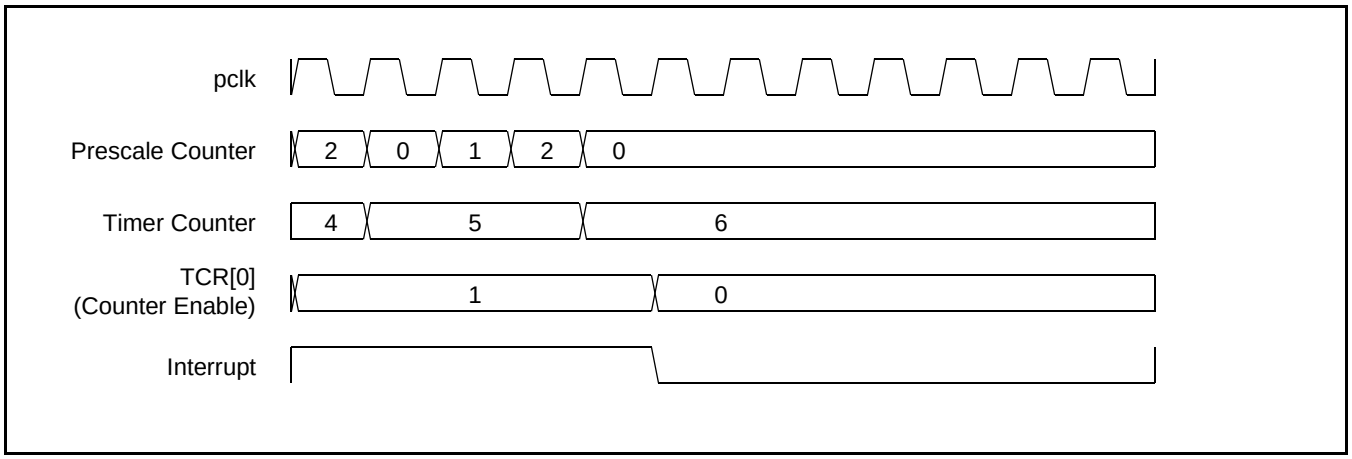


Figure 29: A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled.

ARCHITECTURE

The block diagram for Timer 0 and Timer 1is shown in Figure 30.

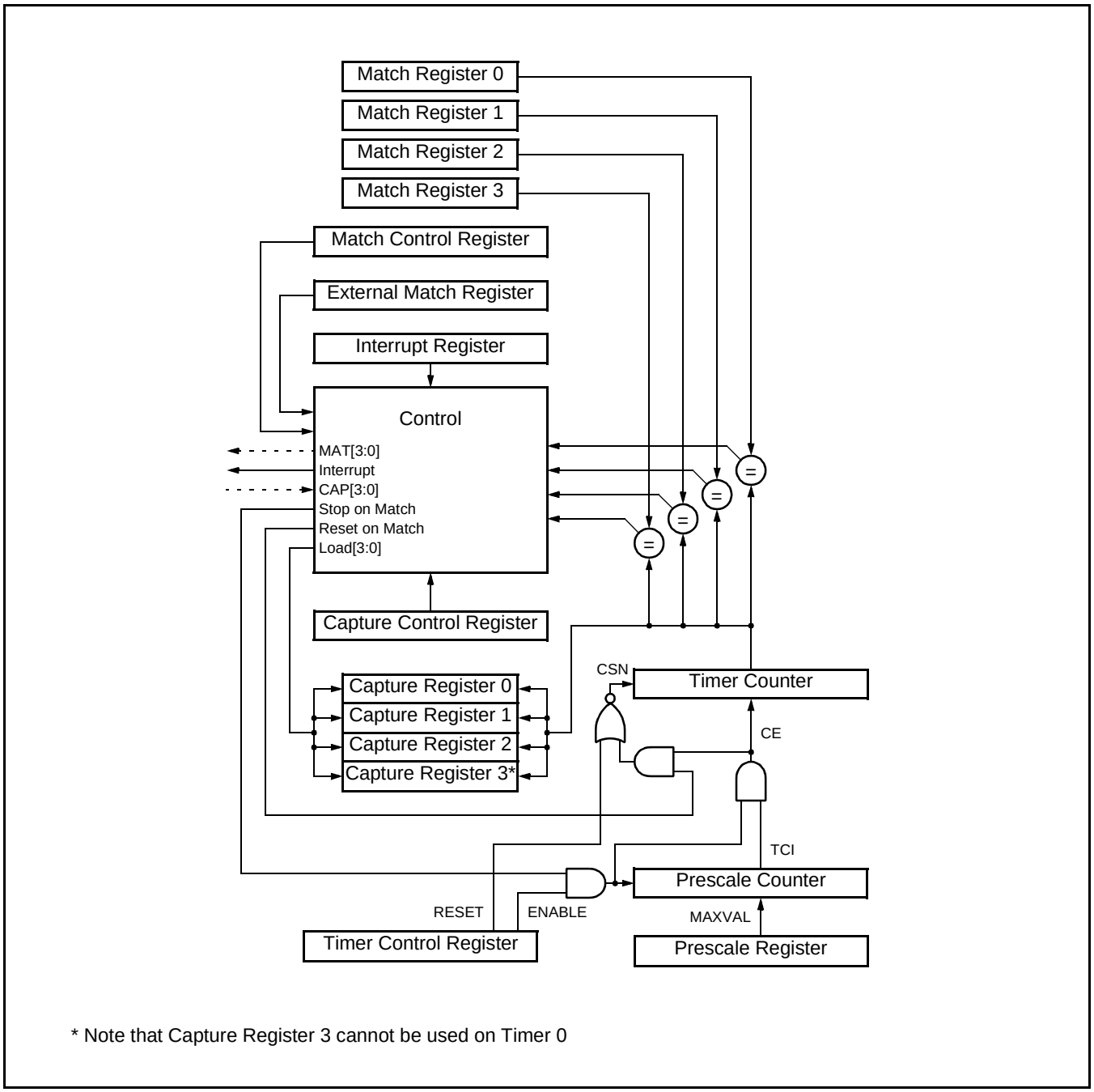


Figure 30: Timer block diagram

14. PULSE WIDTH MODULATOR (PWM)

LPC2106/2105/2104 Pulse Width Modulator is based on standard Timer 0/1 described in previous chapter. Application can choose among PWM and match functions available .

FEATURES

- Seven match registers allow up to 6 single edge controlled or 3 double edge controlled PWM outputs, or a mix of both types. The match registers also allow:
  - Continuous operation with optional interrupt generation on match.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.
- An external output for each match register with the following capabilities:
  - Set low on match.
  - Set high on match.
  - Toggle on match.
  - Do nothing on match.
- Supports single edge controlled and/or double edge controlled PWM outputs. Single edge controlled PWM outputs all go high at the beginning of each cycle unless the output is a constant low. Double edge controlled PWM outputs can have either edge occur at any position within a cycle. This allows for both positive going and negative going pulses.
- Pulse period and width can be any number of timer counts. This allows complete flexibility in the trade-off between resolution and repetition rate. All PWM outputs will occur at the same repetition rate.
- Double edge controlled PWM outputs can be programmed to be either positive going or negative going pulses.
- Match register updates are synchronized with pulse outputs to prevent generation of erroneous pulses. Software must "release" new match values before they can become effective.
- May be used as a standard timer if the PWM mode is not enabled.
- A 32-bit Timer/Counter with a programmable 32-bit Prescaler.
- Four 32-bit capture channels take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt.

DESCRIPTION

The PWM is based on the standard Timer block and inherits all of its features, although only the PWM function is pinned out on the LPC2106/2105/2104. The Timer is designed to count cycles of the peripheral clock (pclk) and optionally generate interrupts or perform other actions when specified timer values occur, based on seven match registers. It also includes four capture inputs to save the timer value when an input signal transitions, and optionally generate an interrupt when those events occur. The PWM function is in addition to these features, and is based on match register events.

The ability to separately control rising and falling edge locations allows the PWM to be used for more applications. For instance, multi-phase motor control typically requires three non-overlapping PWM outputs with individual control of all three pulse widths and positions.

Two match registers can be used to provide a single edge controlled PWM output. One match register (PWMMR0) controls the PWM cycle rate, by resetting the count upon match. The other match register controls the PWM edge position. Additional single edge controlled PWM outputs require only one match register each, since the repetition rate is the same for all PWM outputs. Multiple single edge controlled PWM outputs will all have a rising edge at the beginning of each PWM cycle, when an PWMMR0 match occurs.

Three match registers can be used to provide a PWM output with both edges controlled. Again, the PWMMR0 match register controls the PWM cycle rate. The other match registers control the two PWM edge positions. Additional double edge controlled PWM outputs require only two match registers each, since the repetition rate is the same for all PWM outputs.

With double edge controlled PWM outputs, specific match registers control the rising and falling edge of the output. This allows both positive going PWM pulses (when the rising edge occurs prior to the falling edge), and negative going PWM pulses (when the falling edge occurs prior to the rising edge).

Figure 31 shows the block diagram of the PWM. The portions that have been added to the standard timer block are on the right hand side and at the top of the diagram.

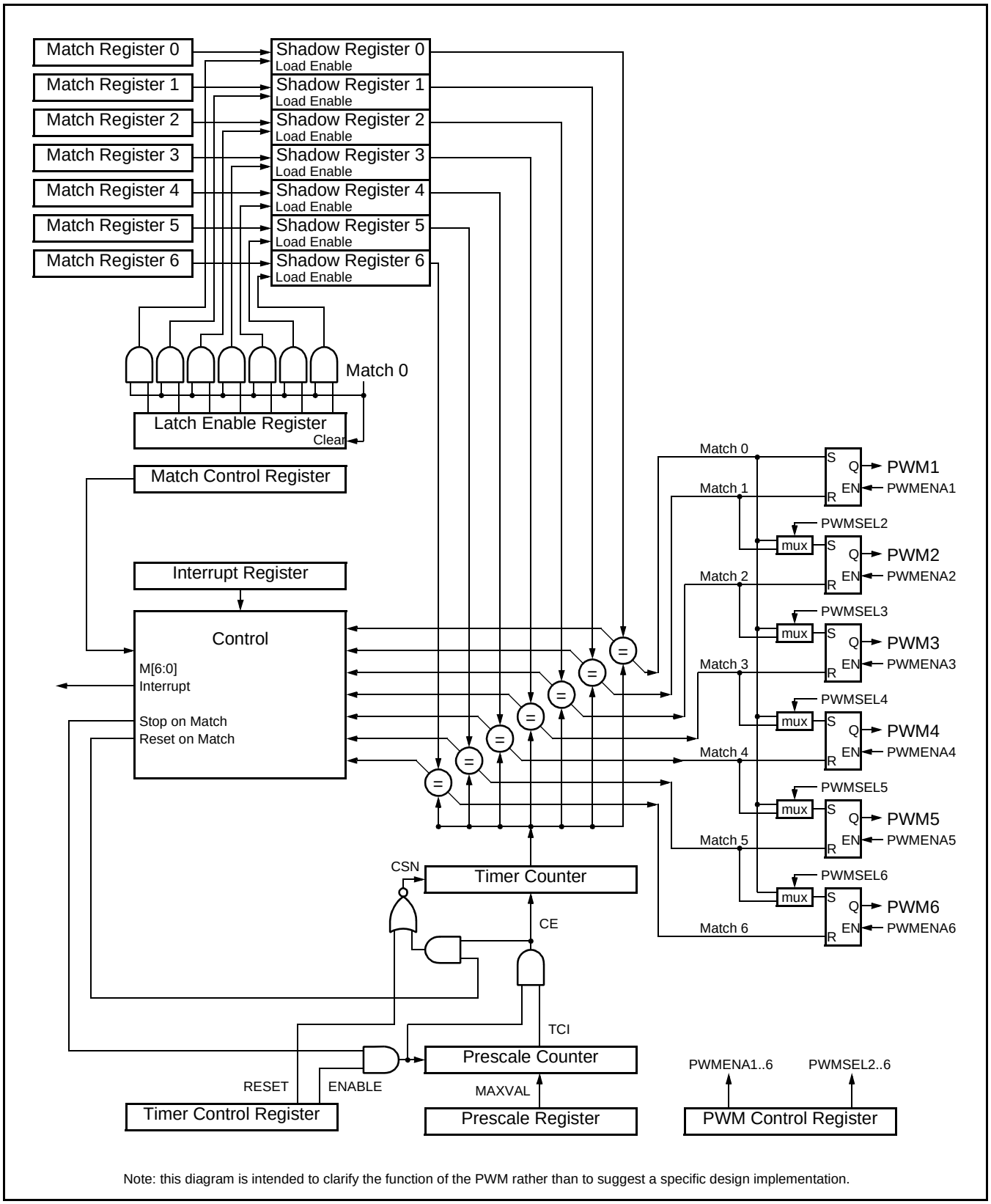


Figure 31: PWM block diagram

A sample of how PWM values relate to waveform outputs is shown in Figure 32. PWM output logic is shown in Figure 31 that allows selection of either single or double edge controlled PWM outputs via the muxes controlled by the PWMSELn bits. The match register selections for various PWM outputs is shown in Table 113. This implementation supports up to N-1 single edge PWM outputs or (N-1)/2 double edge PWM outputs, where N is the number of match registers that are implemented. PWM types can be mixed if desired.

The waveforms below show a single PWM cycle and demonstrate PWM outputs under the following conditions:

- The timer is configured for PWM mode.
- Match 0 is configured to reset the timer/counter when a match event occurs.
- Control bits PWMSEL2 and PWMSEL4 are set.
- The Match register values are as follows:

  MR0= 100(PWM rate)
  MR1= 41, MR2 = 78(PWM2 output)
  MR3= 53, MR4 = 27(PWM4 output)
  MR5= 65(PWM5 output)

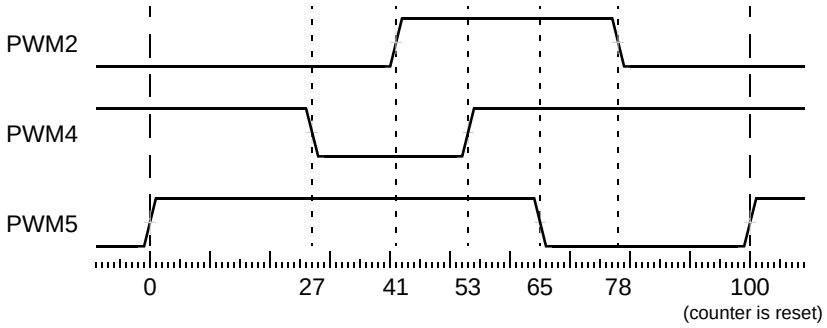


Figure 32: Sample PWM waveforms

Table 113: Set and Reset inputs for PWM Flip-Flops

PWM Channel	Single Edge PWM (PWMSELn = 0)		Double Edge PWM (PWMSELn = 1)	
	Set by	Reset by	Set by	Reset by
1	Match 0	Match 1	Match 0 [1]	Match 1 [1]
2	Match 0	Match 2	Match 1	Match 2
3	Match 0	Match 3	Match 2 [2]	Match 3 [2]
4	Match 0	Match 4	Match 3	Match 4
5	Match 0	Match 5	Match 4 [2]	Match 5 [2]
6	Match 0	Match 6	Match 5	Match 6

Notes:
1. Identical to single edge mode in this case since Match 0 is the neighboring match register. Essentially, PWM1 cannot be a double edged output.
2. It is generally not advantageous to use PWM channels 3 and 5 for double edge PWM outputs because it would reduce the number of double edge PWM outputs that are possible. Using PWM 2, PWM4, and PWM6 for double edge PWM outputs provides the most pairings.

Rules for Single Edge Controlled PWM Outputs

1. All single edge controlled PWM outputs go high at the beginning of a PWM cycle unless their match value is equal to 0.
2. Each PWM output will go low when its match value is reached. If no match occurs (i.e. the match value is greater than the PWM rate), the PWM output remains continuously high.

Rules for Double Edge Controlled PWM Outputs

Five rules are used to determine the next value of a PWM output when a new cycle is about to begin:

1. The match values for the next PWM cycle are used at the end of a PWM cycle (a time point which is coincident with the beginning of the next PWM cycle), except as noted in rule 3.
2. A match value equal to 0 or the current PWM rate (the same as the Match channel 0 value) have the same effect, except as noted in rule 3. For example, a request for a falling edge at the beginning of the PWM cycle has the same effect as a request for a falling edge at the end of a PWM cycle.
3. When match values are changing, if one of the "old" match values is equal to the PWM rate, it is used again once if the neither of the new match values are equal to 0 or the PWM rate, and there was no old match value equal to 0.
4. If both a set and a clear of a PWM output are requested at the same time, clear takes precedence. This can occur when the set and clear match values are the same as in, or when the set or clear value equals 0 and the other value equals the PWM rate.
5. If a match value is out of range (i.e. greater than the PWM rate value), no match event occurs and that match channel has no effect on the output. This means that the PWM output will remain always in one state, allowing always low, always high, or "no change" outputs.

PIN DESCRIPTION

Table 114 gives a brief summary of each of PWM related pins.

Table 114: Pin summary

Pin name	Pin direction	Pin Description
PWM1	Output	Output from PWM channel 1.
PWM2	Output	Output from PWM channel 2.
PWM3	Output	Output from PWM channel 3.
PWM4	Output	Output from PWM channel 4.
PWM5	Output	Output from PWM channel 5.
PWM6	Output	Output from PWM channel 6.

REGISTER DESCRIPTION

The PWM function adds new registers and registers bits as shown in Table 115 below.

Table 115: Pulse Width Modulator Register Map

Address	Name	Description	Access	Reset Value*
0xE0014000	PWMIR	PWM Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of the possible interrupt sources are pending.	R/W	0
0xE0014004	PWMTCR	PWM Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	R/W	0
0xE0014008	PWMTC	PWM Timer Counter. The 32-bit TC is incremented every PR+1 cycles of pclk. The TC is controlled through the TCR.	RW	0
0xE001400C	PWMPR	PWM Prescale Register. The TC is incremented every PR+1 cycles of pclk.	R/W	0
0xE0014010	PWMPC	PWM Prescale Counter. The 32-bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented.	R/W	0
0xE0014014	PWMMCR	PWM Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	R/W	0
0xE0014018	PWMMR0	PWM Match Register 0. MR0 can be enabled through MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt when it matches the TC. In addition, a match between MR0 and the TC sets all PWM outputs that are in single-edge mode, and sets PWM1 if it is in double-edge mode.	R/W	0
0xE001401C	PWMMR1	PWM Match Register 1. MR1 can be enabled through MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt when it matches the TC. In addition, a match between MR1 and the TC clears PWM1 in either single-edge mode or double-edge mode, and sets PWM2 if it is in double-edge mode.	R/W	0
0xE0014020	PWMMR2	PWM Match Register 2. MR2 can be enabled through MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt when it matches the TC. In addition, a match between MR2 and the TC clears PWM2 in either single-edge mode or double-edge mode, and sets PWM3 if it is in double-edge mode.	R/W	0
0xE0014024	PWMMR3	PWM Match Register 3. MR3 can be enabled through MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt when it matches the TC. In addition, a match between MR3 and the TC clears PWM3 in either single-edge mode or double-edge mode, and sets PWM4 if it is in double-edge mode.	R/W	0
0xE0014040	PWMMR4	PWM Match Register 4. MR4 can be enabled through MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt when it matches the TC. In addition, a match between MR4 and the TC clears PWM4 in either single-edge mode or double-edge mode, and sets PWM5 if it is in double-edge mode.	R/W	0

Table 115: Pulse Width Modulator Register Map

Address	Name	Description	Access	Reset Value*
0xE0014044	PWMMR5	PWM Match Register 5. MR5 can be enabled through MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt when it matches the TC. In addition, a match between MR5 and the TC clears PWM5 in either single-edge mode or double-edge mode, and sets PWM6 if it is in double-edge mode.	R/W	0
0xE0014048	PWMMR6	PWM Match Register 6. MR6 can be enabled through MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt when it matches the TC. In addition, a match between MR6 and the TC clears PWM6 in either single-edge mode or double-edge mode.	R/W	0
0xE001404C	PWMPCR	PWM Control Register. Enables PWM outputs and selects PWM channel types as either single edge or double edge controlled.	R/W	0
0xE0014050	PWMLER	PWM Latch Enable Register. Enables use of new PWM match values.	R/W	0

*Reset Value refers to the data stored in used bits only. It does not include reserved bits content.

PWM Interrupt Register (PWMIR - 0xE0014000)

The PWM Interrupt Register consists of eleven bits (Table 116), seven for the match interrupts and four reserved for the future use. If an interrupt is generated then the corresponding bit in the PWMIR will be high. Otherwise, the bit will be low. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect.

Table 116: PWM Interrupt Register (PWMIR - 0xE0014000)

PWMIR	Function	Description	Reset Value
0	PWMMR0 Interrupt	Interrupt flag for PWM match channel 0.	0
1	PWMMR1 Interrupt	Interrupt flag for PWM match channel 1.	0
2	PWMMR2 Interrupt	Interrupt flag for PWM match channel 2.	0
3	MR3 Interrupt	Interrupt flag for PWM match channel 3.	0
4	Reserved.	Application must not write 1 to this bit.	0
5	Reserved.	Application must not write 1 to this bit.	0
6	Reserved.	Application must not write 1 to this bit.	0
7	Reserved.	Application must not write 1 to this bit.	0
8	PWMMR4 Interrupt	Interrupt flag for PWM match channel 4.	0
9	PWMMR5 Interrupt	Interrupt flag for PWM match channel 5.	0
10	PWMMR6 Interrupt	Interrupt flag for PWM match channel 6.	0

PWM Timer Control Register (PWMTCR - 0xE0014004)

The PWM Timer Control Register (PWMTCR) is used to control the operation of the PWM Timer Counter. The function of each of the bits is shown in Table 117.

Table 117: PWM Timer Control Register (PWMTCR - 0xE0014004)

PWMTCR	Function	Description	Reset Value
0	Counter Enable	When one, the PWM Timer Counter and PWM Prescale Counter are enabled for counting. When zero, the counters are disabled.	0
1	Counter Reset	When one, the PWM Timer Counter and the PWM Prescale Counter are synchronously reset on the next positive edge of pclk. The counters remain reset until TCR[1] is returned to zero.	0
2	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	PWM Enable	When one, PWM mode is enabled. PWM mode causes shadow registers to operate in connection with the Match registers. A program write to a Match register will not have an effect on the Match result until the corresponding bit in PWMLER has been set, followed by the occurrence of a PWM Match 0 event. Note that the PWM Match register that determines the PWM rate (PWM Match 0) must be set up prior to the PWM being enabled. Otherwise a Match event will not occur to cause shadow register contents to become effective.	0

PWM Timer Counter (PWMTC - 0xE0014008)

The 32-bit PWM Timer Counter is incremented when the Prescale Counter reaches its terminal count. Unless it is reset before reaching its upper limit, the PWMTC will count up through the value 0xFFFFFFFF and then wrap back to the value 0x00000000. This event does not cause an interrupt, but a Match register can be used to detect an overflow if needed.

PWM Prescale Register (PWMPR - 0xE001400C)

The 32-bit PWM Prescale Register specifies the maximum value for the PWM Prescale Counter.

PWM Prescale Counter Register (PWMPC - 0xE0014010)

The 32-bit PWM Prescale Counter controls division of pclk by some constant value before it is applied to the PWM Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The PWM Prescale Counter is incremented on every pclk. When it reaches the value stored in the PWM Prescale Register, the PWM Timer Counter is incremented and the PWM Prescale Counter is reset on the next pclk. This causes the PWM TC to increment on every pclk when PWMPR = 0, every 2 pclks when PWMPR = 1, etc.

PWM Match Registers (PWMMR0 - PWMMR6)

ThePWM Match register values are continuously compared to the PWM Timer Counter value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the PWM Timer Counter, or stop the timer. Actions are controlled by the settings in the PWMMCR register.

PWM Match Control Register (PWMMCR - 0xE0014014)

The PWM Match Control Register is used to control what operations are performed when one of the PWM Match Registers matches the PWM Timer Counter. The function of each of the bits is shown in Table 118.

Table 118: PWM Match Control Register (PWMMCR - 0xE0014014)

PWMMCR	Function	Description	Reset Value
0	Interrupt on PWMMR0	When one, an interrupt is generated when PWMMR0 matches the value in the PWMTC. When zero this interrupt is disabled.	0
1	Reset on PWMMR0	When one, the PWMTC will be reset if PWMMR0 matches it. When zero this feature is disabled.	0
2	Stop on PWMMR0	When one, the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR0 matches the PWMTC. When zero this feature is disabled.	0
3	Interrupt on PWMMR1	When one, an interrupt is generated when PWMMR1 matches the value in the PWMTC. When zero this interrupt is disabled.	0
4	Reset on PWMMR1	When one, the PWMTC will be reset if PWMMR1 matches it. When zero this feature is disabled.	0
5	Stop on PWMMR1	When one, the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR1 matches the PWMTC. When zero this feature is disabled	0
6	Interrupt on PWMMR2	When one, an interrupt is generated when PWMMR2 matches the value in the PWMTC. When zero this interrupt is disabled.	0
7	Reset on PWMMR2	When one, the PWMTC will be reset if PWMMR2 matches it. When zero this feature is disabled.	0
8	Stop on PWMMR2	When one, the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR2 matches the PWMTC. When zero this feature is disabled	0
9	Interrupt on PWMMR3	When one, an interrupt is generated when PWMMR3 matches the value in the PWMTC. When zero this interrupt is disabled	0
10	Reset on PWMMR3	When one, the PWMTC will be reset if PWMMR3 matches it. When zero this feature is disabled.	0
11	Stop on PWMMR3	When one, the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR3 matches the PWMTC. When zero this feature is disabled	0
12	Interrupt on PWMMR4	When one, an interrupt is generated when PWMMR4 matches the value in the PWMTC. When zero this interrupt is disabled.	0
13	Reset on PWMMR4	When one, the PWMTC will be reset if PWMMR4 matches it. When zero this feature is disabled.	0
14	Stop on PWMMR4	When one, the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR4 matches the PWMTC. When zero this feature is disabled	0
15	Interrupt on PWMMR5	When one, an interrupt is generated when PWMMR5 matches the value in the PWMTC. When zero this interrupt is disabled.	0
16	Reset on PWMMR5	When one, the PWMTC will be reset if PWMMR5 matches it. When zero this feature is disabled.	0

Table 118: PWM Match Control Register (PWMMCR - 0xE0014014)

PWMMCR	Function	Description	Reset Value
17	Stop on PWMMR5	When one, the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR5 matches the PWMTC. When zero this feature is disabled	0
18	Interrupt on PWMMR6	When one, an interrupt is generated when PWMMR6 matches the value in the PWMTC. When zero this interrupt is disabled.	0
19	Reset on PWMMR6	When one, the PWMTC will be reset if PWMMR6 matches it. When zero this feature is disabled.	0
20	Stop on PWMMR6	When one, the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR6 matches the PWMTC. When zero this feature is disabled	0

PWM Control Register (PWMPCR - 0xE001404C)

The PWM Control Register is used to enable and select the type of each PWM channel. The function of each of the bits are shown in Table 119.

Table 119: PWM Control Register (PWMPCR - 0xE001404C)

PWMPCR	Function	Description	Reset Value
1:0	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	PWMSEL2	When zero, selects single edge controlled mode for PWM2. When one, selects double edge controlled mode for the PWM2 output.	0
3	PWMSEL3	When zero, selects single edge controlled mode for PWM3. When one, selects double edge controlled mode for the PWM3 output.	0
4	PWMSEL4	When zero, selects single edge controlled mode for PWM4. When one, selects double edge controlled mode for the PWM4 output.	0
5	PWMSEL5	When zero, selects single edge controlled mode for PWM5. When one, selects double edge controlled mode for the PWM5 output.	0
6	PWMSEL6	When zero, selects single edge controlled mode for PWM6. When one, selects double edge controlled mode for the PWM6 output.	0
8:7	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
9	PWMENA1	When one, enables the PWM1 output. When zero, disables the PWM1 output.	0
10	PWMENA2	When one, enables the PWM2 output. When zero, disables the PWM2 output.	0
11	PWMENA3	When one, enables the PWM3 output. When zero, disables the PWM3 output.	0
12	PWMENA4	When one, enables the PWM4 output. When zero, disables the PWM4 output.	0
13	PWMENA5	When one, enables the PWM5 output. When zero, disables the PWM5 output.	0
14	PWMENA6	When one, enables the PWM6 output. When zero, disables the PWM6 output.	0
15	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

PWM Latch Enable Register (PWMLER - 0xE0014050)

ThePWM Latch Enable Register is used to control the update of the PWM Match registers when they are used for PWM generation. When software writes to the location of a PWM Match register while the Timer is in PWM mode, the value is held in a shadow register. When a PWM Match 0 event occurs (normally also resetting the timer in PWM mode), the contents of shadow registers will be transferred to the actual Match registers if the corresponding bit in the Latch Enable Register has been set. At that point, the new values will take effect and determine the course of the next PWM cycle. Once the transfer of new values has taken place, all bits of the LER are automatically cleared. Until the corresponding bit in the PWMLER is set and a PWM Match 0 event occurs, any value written to the PWM Match registers has no effect on PWM operation.

For example, if PWM2 is configured for double edge operation and is currently running, a typical sequence of events for changing the timing would be:

- Write a new value to the PWM Match1 register.
- Write a new value to the PWM Match2 register.
- Write to the PWMLER, setting bits 1 and 2 at the same time.
- The altered values will become effective at the next reset of the timer (when a PWM Match 0 event occurs).

The order of writing the two PWM Match registers is not important, since neither value will be used until after the write to PWMLER. This insures that both values go into effect at the same time, if that is required. A single value may be altered in the same way if needed.

The function of each of the bits in the PWMLER is shown in Table 120.

Table 120: PWM Latch Enable Register (PWMLER - 0xE0014050)

PWMLER	Function	Description	Reset Value
0	Enable PWM Match 0 Latch	Writing a one to this bit allows the last value written to the PWM Match 0 register to be become effective when the timer is next reset by a PWM Match event. See the description of the PWM Match Control Register (PWMMCR).	0
1	Enable PWM Match 1 Latch	Writing a one to this bit allows the last value written to the PWM Match 1 register to be become effective when the timer is next reset by a PWM Match event. See the description of the PWM Match Control Register (PWMMCR).	0
2	Enable PWM Match 2 Latch	Writing a one to this bit allows the last value written to the PWM Match 2 register to be become effective when the timer is next reset by a PWM Match event. See the description of the PWM Match Control Register (PWMMCR).	0
3	Enable PWM Match 3 Latch	Writing a one to this bit allows the last value written to the PWM Match 3 register to be become effective when the timer is next reset by a PWM Match event. See the description of the PWM Match Control Register (PWMMCR).	0
4	Enable PWM Match 4 Latch	Writing a one to this bit allows the last value written to the PWM Match 4 register to be become effective when the timer is next reset by a PWM Match event. See the description of the PWM Match Control Register (PWMMCR).	0
5	Enable PWM Match 5 Latch	Writing a one to this bit allows the last value written to the PWM Match 5 register to be become effective when the timer is next reset by a PWM Match event. See the description of the PWM Match Control Register (PWMMCR).	0
6	Enable PWM Match 6 Latch	Writing a one to this bit allows the last value written to the PWM Match 6 register to be become effective when the timer is next reset by a PWM Match event. See the description of the PWM Match Control Register (PWMMCR).	0
7	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

15. REAL TIME CLOCK

FEATURES

- Measures the passage of time to maintain a calendar and clock.
- Ultra Low Power design to support battery powered systems.
- Provides Seconds, Minutes, Hours, Day of Month, Month, Year, Day of Week, and Day of Year.
- Programmable Reference Clock Divider allows adjustment of the RTC to match various crystal frequencies.

DESCRIPTION

The Real Time Clock (RTC) is designed to provide a set of counters to measure time during system power on and off operation. The RTC has been designed to use little power, making it suitable for battery powered systems where the CPU is not running continuously (Idle mode).

ARCHITECTURE

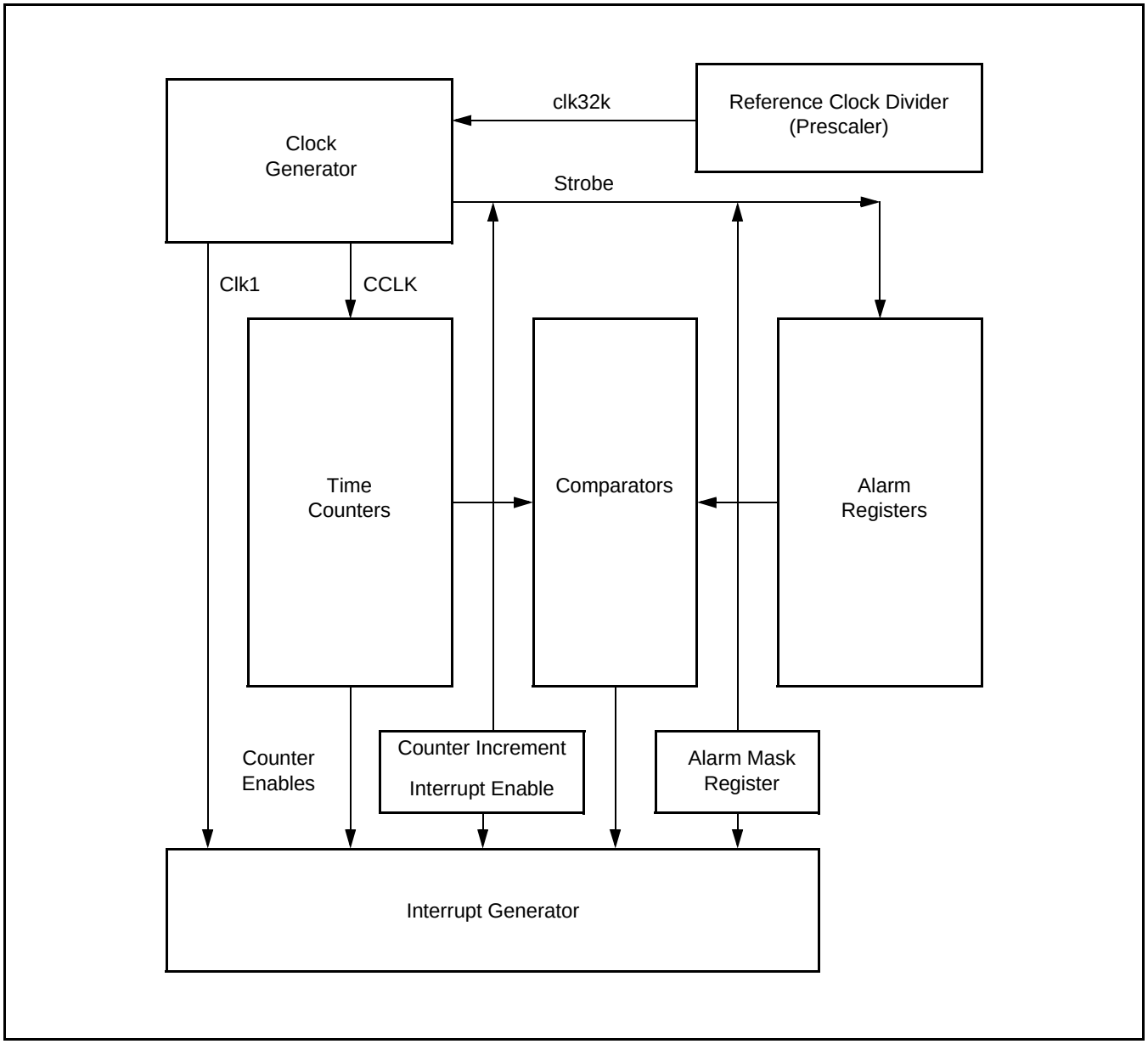


Figure 33: RTC block diagram

REGISTER DESCRIPTION

The RTC includes a number of registers. The address space is split into four sections by functionality. The first eight addresses are the Miscellaneous Register Group. The second set of eight locations are the Time Counter Group. The third set of eight locations contain the Alarm Register Group. The remaining registers control the Reference Clock Divider.

The Real Time Clock includes the register shown in Table 121. Detailed descriptions of the registers follow.

Table 121: Real Time Clock Register Map

Address	Name	Size	Description	Access	Reset Value
0xE0024000	ILR	2	Interrupt Location Register	R/W	*
0xE0024004	CTC	15	Clock Tick Counter.	RO	*
0xE0024008	CCR	4	Clock Control Register	R/W	*
0xE002400C	CIIR	8	Counter Increment Interrupt Register	R/W	*
0xE0024010	AMR	8	Alarm Mask Register	R/W	*
0xE0024014	CTIME0	(32)	Consolidated Time Register 0	RO	*
0xE0024018	CTIME1	(32)	Consolidated Time Register 1	RO	*
0xE002401C	CTIME2	(32)	Consolidated Time Register 2	RO	*
0xE0024020	SEC	6	Seconds Register	R/W	*
0xE0024024	MIN	6	Minutes Register	R/W	*
0xE0024028	HOUR	5	Hours Register	R/W	*
0xE002402C	DOM	5	Day of Month Register	R/W	*
0xE0024030	DOW	3	Day of Week Register	R/W	*
0xE0024034	DOY	9	Day of Year Register	R/W	*
0xE0024038	MONTH	4	Months Register	R/W	*
0xE002403C	YEAR	12	Years Register	R/W	*
0xE0024060	ALSEC	6	Alarm value for Seconds	R/W	*
0xE0024064	ALMIN	6	Alarm value for Minutes	R/W	*
0xE0024068	ALHOUR	5	Alarm value for Hours	R/W	*
0xE002406C	ALDOM	5	Alarm value for Day of Month	R/W	*
0xE0024070	ALDOW	3	Alarm value for Day of Week	R/W	*
0xE0024074	ALDOY	9	Alarm value for Day of Year	R/W	*
0xE0024078	ALMON	4	Alarm value for Months	R/W	*
0xE002407C	ALYEAR	12	Alarm value for Year	R/W	*
0xE0024080	PREINT	13	Prescale value, integer portion	R/W	0
0xE0024084	PREFRAC	15	Prescale value, fractional portion	R/W	0

* Registers in the RTC other than those that are part of the Prescaler are not affected by chip Reset. These registers must be initialized by software if the RTC is enabled.

RTC INTERRUPTS

Interrupt generation is controlled through the Interrupt Location Register (ILR), Counter Increment Interrupt Register (CIIR), the alarm registers, and the Alarm Mask Register (AMR). Interrupts are generated only by the transition into the interrupt state. The ILR separately enables CIIR and AMR interrupts. Each bit in CIIR corresponds to one of the time counters. If CIIR is enabled for a particular counter, then every time the counter is incremented an interrupt is generated. The alarm registers allow the user to specify a date and time for an interrupt to be generated. The AMR provides a mechanism to mask alarm compares. If all non-masked alarm registers match the value in their corresponding time counter, then an interrupt is generated.

MISCELLANEOUS REGISTER GROUP

Table 122 summarizes the registers located from 0 to 7 of A[6:2]. More detailed descriptions follow.

Table 122: Miscellaneous Registers

Address	Name	Size	Description	Access
0xE0024000	ILR	2	Interrupt Location. Reading this location indicates the source of an interrupt. Writing a one to the appropriate bit at this location clears the associated interrupt.	RW
0xE0024004	CTC	15	Clock Tick Counter. Value from the clock divider.	RO
0xE0024008	CCR	4	Clock Control Register. Controls the function of the clock divider.	RW
0xE002400C	CIIR	8	Counter Increment Interrupt. Selects which counters will generate an interrupt when they are incremented.	RW
0xE0024010	AMR	8	Alarm Mask Register. Controls which of the alarm registers are masked.	RW
0xE0024014	CTIME0	32	Consolidated Time Register 0	RO
0xE0024018	CTIME1	32	Consolidated Time Register 1	RO
0xE002401C	CTIME2	32	Consolidated Time Register 2	RO

Interrupt Location (ILR - 0xE0024000)

The Interrupt Location Register is a 2-bit register that specifies which blocks are generating an interrupt (see Table 123). Writing a one to the appropriate bit clears the corresponding interrupt. Writing a zero has no effect. This allows the programmer to read this register and write back the same value to clear only the interrupt that is detected by the read.

Table 123: Interrupt Location Register Bits (ILR - 0xE0024000)

ILR	Function	Description
0	RTCCIF	When one, the Counter Increment Interrupt block generated an interrupt. Writing a one to this bit location clears the counter increment interrupt.
1	RTCALF	When one, the alarm registers generated an interrupt. Writing a one to this bit location clears the alarm interrupt.

Clock Tick Counter (CTC - 0xE0024004)

The Clock Tick Counter is read only. It can be reset to zero through the Clock Control Register (CCR). The CTC consists of the bits of the clock divider counter.

Table 124: Clock Tick Counter Bits (CTC - 0xE0024004)

CTC	Function	Description
0	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
15:1	Clock Tick Counter	Prior to the Seconds counter, the CTC counts 32,768 clocks per second. Due to the RTC Prescaler, these 32,768 time increments may not all be of the same duration. Refer to the Reference Clock Divider (Prescaler) description for details.

Clock Control Register (CCR - 0xE0024008)

The clock register is a 4-bit register that controls the operation of the clock divide circuit. Each bit of the clock register is described in Table 125.

Table 125: Clock Control Register Bits (CCR - 0xE0024008)

CCR	Function	Description
0	CLKEN	Clock Enable. When this bit is a one the time counters are enabled. When it is a zero, they are disabled so that they may be initialized.
1	CTCRST	CTC Reset. When one, the elements in the Clock Tick Counter are reset. The elements remain reset until CCR[1] is changed to zero.
3:2	CTTEST	Test Enable. These bits should always be zero during normal operation.

Counter Increment Interrupt

The Counter Increment Interrupt Register (CIIR) gives the ability to generate an interrupt every time a counter is incremented. This interrupt remains valid until cleared by writing a one to bit zero of the Interrupt Location Register (ILR[0]).

Table 126: Counter Increment Interrupt Register Bits (CIIR - 0xE002400C)

CIIR	Function	Description
0	IMSEC	When one, an increment of the Second value generates an interrupt.
1	IMMIN	When one, an increment of the Minute value generates an interrupt.
2	IMHOUR	When one, an increment of the Hour value generates an interrupt.
3	IMDOM	When one, an increment of the Day of Month value generates an interrupt.
4	IMDOW	When one, an increment of the Day of Week value generates an interrupt.
5	IMDOY	When one, an increment of the Day of Year value generates an interrupt.
6	IMMON	When one, an increment of the Month value generates an interrupt.
7	IMYEAR	When one, an increment of the Year value generates an interrupt.

Alarm Mask

The Alarm Mask Register (AMR) allows the user to mask any of the alarm registers. Table 127 shows the relationship between the bits in the AMR and the alarms. For the alarm function, every non-masked alarm register must match the corresponding time counter for an interrupt to be generated. The interrupt is generated only when the counter comparison first changes from no match to match. The interrupt is removed when a one is written to the appropriate bit of the Interrupt Location Register (ILR). If all mask bits are set, then the alarm is disabled.

Table 127: Alarm Mask Register Bits (AMR - 0xE0024010)

AMR	Function	Description
0	AMRSEC	When one, the Second value is not compared for the alarm.
1	AMRMIN	When one, the Minutes value is not compared for the alarm.
2	AMRHOUR	When one, the Hour value is not compared for the alarm.
3	AMRDOM	When one, the Day of Month value is not compared for the alarm.
4	AMRDOW	When one, the Day of Week value is not compared for the alarm.
5	AMRDOY	When one, the Day of Year value is not compared for the alarm.
6	AMRMON	When one, the Month value is not compared for the alarm.
7	AMRYEAR	When one, the Year value is not compared for the alarm.

CONSOLIDATED TIME REGISTERS

The values of the Time Counters can optionally be read in a consolidated format which allows the programmer to read all time counters with only three read operations. The various registers are packed into 32-bit values as shown in Tables 128, 129, and 130. The least significant bit of each register is read back at bit 0, 8, 16, or 24.

The Consolidated Time Registers are read only. To write new values to the Time Counters, the Time Counter addresses should be used.

Consolidated Time Register 0 (CTIME0 - 0xE0024014)

The Consolidated Time Register 0 contains the low order time values: Seconds, Minutes, Hours, and Day of Week.

Table 128: Consolidated Time Register 0 Bits (CTIME0 - 0xE0024014)

CTIME0	Function	Description
31:27	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
26:24	Day of Week	Day of week value in the range of 0 to 6.
23:21	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
20:16	Hours	Hours value in the range of 0 to 23.
15:14	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
13:8	Minutes	Minutes value in the range of 0 to 59.
7:6	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
5:0	Seconds	Seconds value in the range of 0 to 59.

Consolidated Time Register 1 (CTIME1 - 0xE0024018)

The Consolidate Time Register 1 contains the Day of Month, Month, and Year values.

Table 129: Consolidated Time Register 1 Bits (CTIME1 - 0xE0024018)

CTIME1	Function	Description
31:28	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
27:16	Year	Year value in the range of 0 to 4095.
15:12	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
11:8	Month	Month value in the range of 1 to 12.
7:5	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
4:0	Day of Month	Day of month value in the range of 1 to 28, 29, 30, or 31 (depending on the month and whether it is a leap year).

Consolidated Time Register 2 (CTIME2 - 0xE002401C)

The Consolidate Time Register 2 contains just the Day of Year value.

Table 130: Consolidated Time Register 2 Bits (CTIME2 - 0xE002401C)

CTIME2	Function	Description
11:0	Day of Year	Day of year value in the range of 1 to 365 (366 for leap years).
31:12	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

TIME COUNTER GROUP

The time value consists of the eight counters shown in Tables 131 and 132. These counters can be read or written at the locations shown in Table 132.

Table 131: Time Counter Relationships and Values

Counter	Size	Enabled by	Min value	Maximum value
Second	6	Clk1 (see Figure 33)	0	59
Minute	6	Second	0	59
Hour	5	Minute	0	23
Day of Month	5	Hour	1	28,29,30, or 31
Day of Week	3	Hour	0	6
Day of Year	9	Hour	1	365 or 366 (for leap year)
Month	4	Day of Month	1	12
Year	12	Month or Day of Year	0	4095

Table 132: Time Counter registers

Address	Name	Size	Description	Access
0xE0024020	SEC	6	Seconds value in the range of 0 to 59.	R/W
0xE0024024	MIN	6	Minutes value in the range of 0 to 59.	R/W
0xE0024028	HOUR	5	Hours value in the range of 0 to 23.	R/W
0xE002402C	DOM	5	Day of month value in the range of 1 to 28, 29, 30, or 31 (depending on the month and whether it is a leap year). [1]	R/W
0xE0024030	DOW	3	Day of week value in the range of 0 to 6. [1]	R/W
0xE0024034	DOY	9	Day of year value in the range of 1 to 365 (366 for leap years). [1]	R/W
0xE0024038	MONTH	4	Month value in the range of 1 to 12.	R/W
0xE002403C	YEAR	12	Year value in the range of 0 to 4095.	R/W

Notes:
1. These values are simply incremented at the appropriate intervals and reset at the defined overflow point. They are not calculated and must be correctly initialized in order to be meaningful.

Leap Year Calculation

The RTC does a simple bit comparison to see if the two lowest order bits of the year counter are zero. If true, then the RTC considers that year a leap year. The RTC considers all years evenly divisible by 4 as leap years. This algorithm is accurate from the year 1901 through the year 2099, but fails for the year 2100, which is not a leap year. The only effect of leap year on the RTC is to alter the length of the month of February for the month, day of month, and year counters.

ALARM REGISTER GROUP

The alarm registers are shown in Table 133. The values in these registers are compared with the time counters. If all the unmasked (See “Alarm Mask” on page 162.) alarm registers match their corresponding time counters then an interrupt is generated. The interrupt is cleared when a one is written to bit one of the Interrupt Location Register (ILR[1]).

Table 133: Alarm Registers

Address	Name	Size	Description	Access
0xE0024060	ALSEC	6	Alarm value for Seconds	R/W
0xE0024064	ALMIN	6	Alarm value for Minutes	R/W
0xE0024068	ALHOUR	5	Alarm value for Hours	R/W
0xE002406C	ALDOM	5	Alarm value for Day of Month	R/W
0xE0024070	ALDOW	3	Alarm value for Day of Week	R/W
0xE0024074	ALDOY	9	Alarm value for Day of Year	R/W
0xE0024078	ALMON	4	Alarm value for Months	R/W
0xE002407C	ALYEAR	12	Alarm value for Years	R/W

RTC USAGE NOTES

Since the RTC operates from the VPB clock (pclk), any interruption of that clock will cause the time to drift away from the time value it would have provided otherwise. The variance could be to actual clock time if the RTC was initialized to that, or simply an error in elapsed time since the RTC was activated.

No provision is made in the LPC2106/2105/2104 to retain RTC status upon power loss, or to maintain time incrementation if the clock source is lost, interrupted, or altered. Loss of chip power will result in complete loss of all RTC register contents. Entry to Power Down mode will cause a lapse in the time update. Altering the RTC timebase during system operation (by reconfiguring the PLL, the VPB timer, or the RTC prescaler) will result in some form of accumulated time error.

REFERENCE CLOCK DIVIDER (PRESCALER)

The reference clock divider (hereafter referred to as the Prescaler) allows generation of a 32.768 kHz reference clock from any peripheral clock frequency greater than or equal to 65.536 kHz (2 x 32.768 kHz). This permits the RTC to always run at the proper rate regardless of the peripheral clock rate. Basically, the Prescaler divides the peripheral clock (pclk) by a value which contains both an integer portion and a fractional portion. The result is not a continuous output at a constant frequency, some clock periods will be one pclk longer than others. However, the overall result can always be 32,768 counts per second.

The reference clock divider consists of a 13-bit integer counter and a 15-bit fractional counter. The reasons for these counter sizes are as follows:

1. For frequencies that are expected to be supported by the LPC2106/2105/2104, a 13-bit integer counter is required. This can be calculated as 160 MHz divided by 32,768 minus 1 = 4881 with a remainder of 26,624. Thirteen bits are needed to hold the value 4881, but actually supports frequencies up to 268.4 MHz (32,768 x 8192).
2. The remainder value could be as large as 32,767, which requires 15 bits.

Table 134: Reference Clock Divider registers

Address	Name	Size	Description	Access
0xE0024080	PREINT	13	Prescale Value, integer portion	R/W
0xE0024084	PREFRAC	15	Prescale Value, fractional portion	R/W

Prescaler Integer Register (PREINT - 0xE0024080)

This is the integer portion of the prescale value, calculated as:

PREINT = int (pclk / 32768) - 1. The value of PREINT must be greater than or equal to 1.

Table 135: Prescaler Integer Register (PREINT - 0xE0024080)

PREINT	Function	Description	Reset Value
15:13	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
12:0	Prescaler Integer	Contains the integer portion of the RTC prescaler value.	0

Prescaler Fraction Register (PREFRAC - 0xE0024084)

This is the fractional portion of the prescale value, and may be calculated as:

PREFRAC = pclk - ((PREINT +1) x 32768).

Table 136: Prescaler Fraction Register (PREFRAC - 0xE0024084)

PREFRAC	Function	Description	Reset Value
15	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
14:0	Prescaler Fraction	Contains the fractional portion of the RTC prescaler value.	0

Example of Prescaler Usage

In a simplistic case, the pclk frequency is 65.537 kHz. So:

PREINT = int (pclk / 32768) - 1 = 1 and PREFRAC = pclk - ((PREINT +1) x 32768) = 1

With this prescaler setting, exactly 32,768 clocks per second will be provided to the RTC by counting 2 pclks 32,767 times, and 3 pclks once.

In a more realistic case, the pclk frequency is 10 MHz. Then,

PREINT = int (pclk / 32768) - 1 = 304 and PREFRAC = pclk - ((PREINT +1) x 32768) = 5,760.

In this case, 5,760 of the prescaler output clocks will be 306 (305+1) pclks long, the rest will be 305 pclks long.

In a similar manner, any pclk rate greater than 65.536 kHz (as long as it is an even number of cycles per second) may be turned into a 32 kHz reference clock for the RTC. The only caveat is that if PREFRAC does not contain a zero, then not all of the 32,768 per second clocks are of the same length. Some of the clocks are one pclk longer than others. While the longer pulses are distributed as evenly as possible among the remaining pulses, this "jitter" could possibly be of concern in an application that wishes to observe the contents of the Clock Tick Counter (CTC) directly.

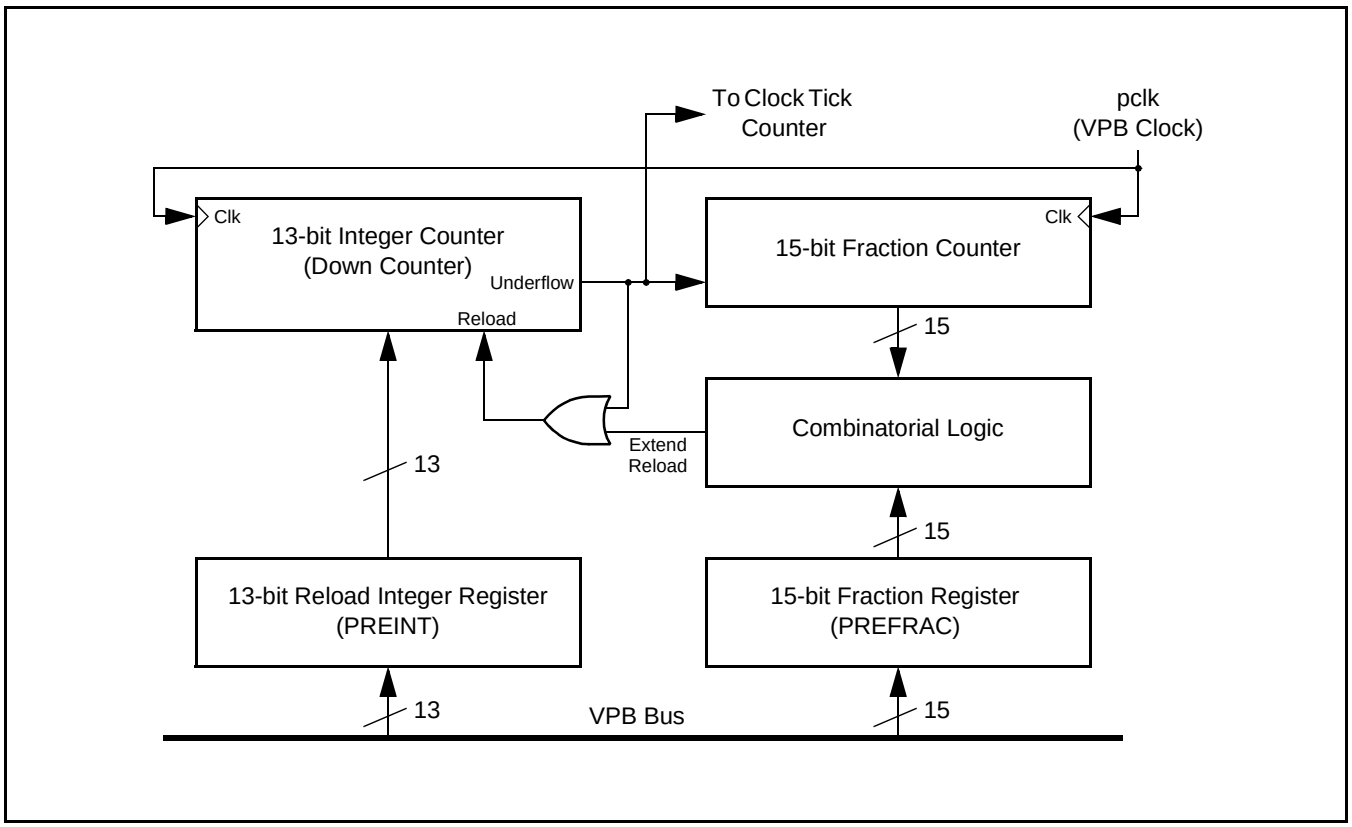


Figure 34: RTC Prescaler block diagram

16. WATCHDOG

FEATURES

- Internally resets chip if not periodically reloaded
- Debug mode
- Enabled by software but requires a hardware reset or a Watchdog reset/interrupt to be disabled
- Incorrect/Incomplete feed sequence causes reset/interrupt if enabled
- Flag to indicate Watchdog reset
- Programmable 32-bit timer with internal pre-scaler
- Selectable time period from (t_{pclk} x 256 x 4) to (t_{pclk} x 2^{32} x 4) in multiples of t_{pclk} x 4

APPLICATIONS

The purpose of the Watchdog is to reset the microcontroller within a reasonable amount of time if it enters an erroneous state. When enabled, the Watchdog will generate a system reset if the user program fails to "feed" (or reload) the Watchdog within a predetermined amount of time.

DESCRIPTION

The Watchdog consists of a divide by 4 fixed pre-scaler and a 32-bit counter. The clock is fed to the timer via a pre-scaler. The timer decrements when clocked. The minimum value from which the counter decrements is 0xFF. Setting a value lower than 0xFF causes 0xFF to be loaded in the counter. Hence the minimum Watchdog interval is (t_{pclk} x 256 x 4) and the maximum Watchdog interval is (t_{pclk} x 2^{32} x 4) in multiples of (t_{pclk} x 4). The Watchdog should be used in the following manner:

- Set the Watchdog timer constant reload value in WDTC register.
- Setup mode in WDMOD register.
- Start the Watchdog by writing 0xAA followed by 0x55 to the WDFEED register.
- Watchdog should be fed again before the Watchdog counter underflows to prevent reset/interrupt.

When the Watchdog counter underflows, the program counter will start from 0x00000000 as in the case of external reset. The Watchdog time-out flag (WDTOF) can be examined to determine if the Watchdog has caused the reset condition. The WDTOF flag must be cleared by software.

REGISTER DESCRIPTION

The Watchdog contains 4 registers as shown in Table 137 below.

Table 137: Watchdog Register Map

Address	Name	Description	Access	Reset Value*
0xE0000000	WDMOD	Watchdog mode register. This register contains the basic mode and status of the Watchdog Timer.	Read/Set	0
0xE0000004	WDTC	Watchdog timer constant register. This register determines the time-out value.	Read/Write	0xFF
0xE0000008	WDFEED	Watchdog feed sequence register. Writing AAh followed by 55h to this register reloads the Watchdog timer to its preset value.	Write Only	NA
0xE000000C	WDTV	Watchdog timer value register. This register reads out the current value of the Watchdog timer.	Read Only	0xFF

*Reset Value refers to the data stored in used bits only. It does not include reserved bits content.

Watchdog Mode Register (WDMOD - 0xE0000000)

The WDMOD register controls the operation of the Watchdog as per the combination of WDEN and RESET bits.

WDEN	WDRESET	
0	X	Debug/Operate without the Watchdog running
1	0	Debug with the Watchdog interrupt but no WDRESET
1	1	Operate with the Watchdog interrupt and WDRESET

Once the WDEN and/or WDRESET bits are set they can not be cleared by software. Both flags are cleared by an external reset or a Watchdog timer underflow.

WDTOF The Watchdog time-out flag is set when the Watchdog times out. This flag is cleared by software.

WDINT The Watchdog interrupt flag is set when the Watchdog times out. This flag is cleared when any reset occurs.

Table 138: Watchdog Mode Register (WDMOD - 0xE0000000)

WDMOD	Function	Description	Reset Value
0	WDEN	Watchdog interrupt enable bit (Set only)	0
1	WDRESET	Watchdog reset enable bit (Set Only)	0
2	WDTOF	Watchdog time-out flag	0 (Only after external reset)
3	WDINT	Watchdog interrupt flag (Read Only)	0
7:4	Reserved	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Watchdog Timer Constant Register (WDTC - 0xE0000004)

The WDTC register determines the time-out value. Every time a feed sequence occurs the WDTC content is reloaded in to the Watchdog timer. It's a 32-bit register with 8 LSB set to 1 on reset. Writing values below 0xFF will cause 0xFF to be loaded to the WDTC. Thus the minimum time-out interval is t_{pclk} x 256 x 4.

WDTC	Function	Description	Reset Value
31:0	Count	Watchdog time-out interval	0xFF

Watchdog Feed Register (WDFEED - 0xE0000008)

Writing 0xAA followed by 0x55 to this register will reload the Watchdog timer to the WDTC value. This operation will also start the Watchdog if it is enabled via the WDMOD register. Setting the WDEN bit in the WDMOD register is not sufficient to enable the Watchdog. A valid feed sequence must first be completed before the Watchdog is capable of generating an interrupt/reset. Until then, the Watchdog will ignore feed errors. Once 0xAA is written to the WDFEED register the next operation in the Watchdog register space should be a **WRITE** (0x55) to the WDFFED register otherwise the Watchdog is triggered. The interrupt/reset will be generated during the second **pclk** following an incorrect access to a watchdog timer register during a feed sequence.

Table 139: Watchdog Feed Register (WDFEED - 0xE0000008)

WDFEED	Function	Description	Reset Value
7:0	Feed	Feed value should be 0xAA followed by 0x55	undefined

Watchdog Timer Value Register (WDTV - 0xE000000C)

The WDTV register is used to read the current value of Watchdog timer.

Table 140: Watchdog Timer Value Register (WDTV - 0xE000000C)

WDTV	Function	Description	Reset Value
31:0	Count	Current timer value	0xFF

BLOCK DIAGRAM

The block diagram of the Watchdog is shown below in the Figure 35.

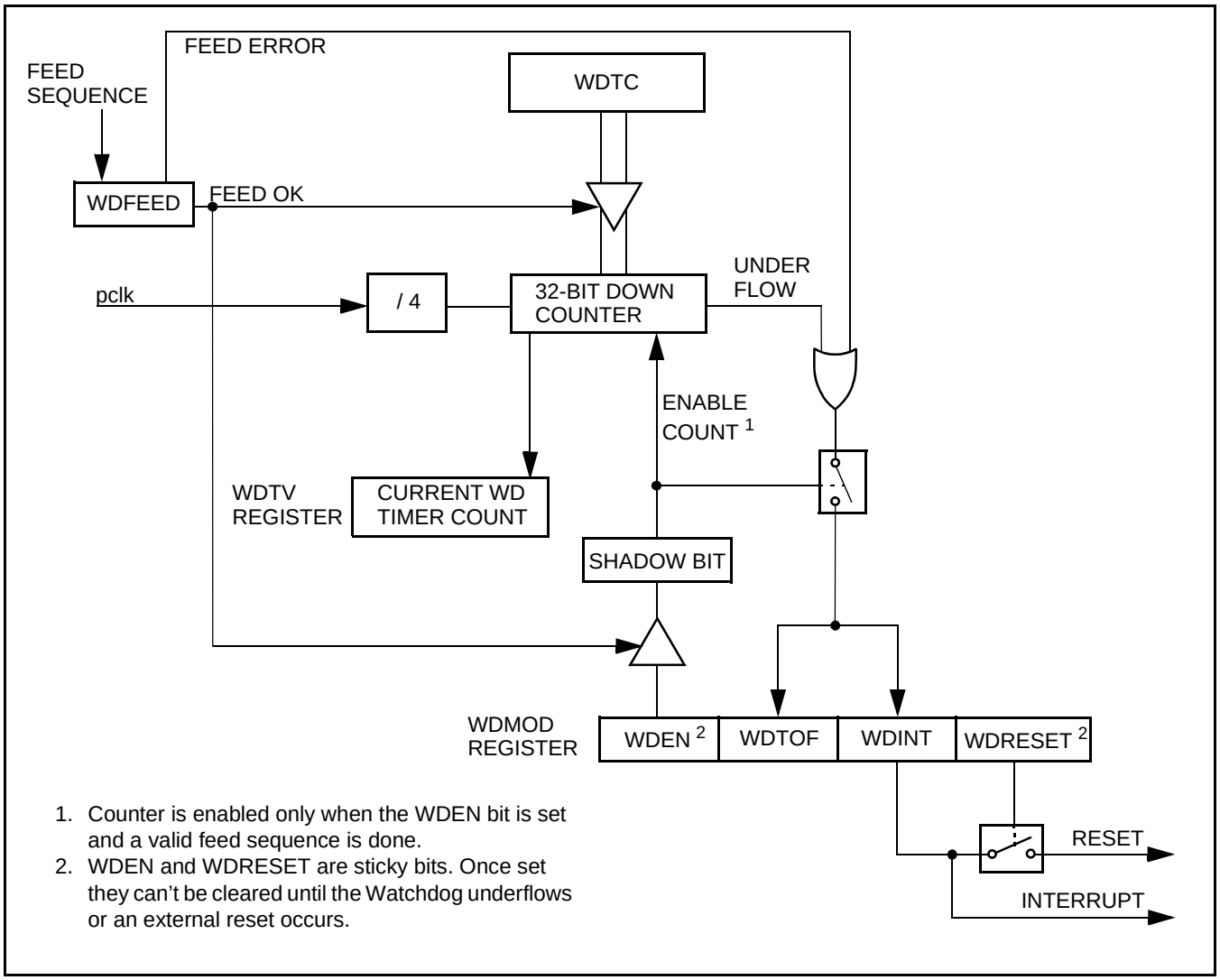


Figure 35: Watchdog Block Diagram

17. FLASH MEMORY SYSTEM AND PROGRAMMING

This chapter describes the Flash Memory System and the Boot Loader.

FLASH MEMORY SYSTEM

The Flash Memory System contains 16 sectors, each of which is 8K bytes in size. Flash memory begins at address 0 and continues upward. Details may be found in the LPC2106/2105/2104 Memory Addressing chapter.

FLASH BOOT LOADER

The Boot Loader controls initial operation after reset, and also provides the means to accomplish programming of the Flash memory. This could be initial programming of a blank device, erasure and re-programming of a previously programmed device, or programming of the Flash memory by the application program in a running system.

FEATURES

- In-System Programming: In-System programming (**ISP**) is programming as well as reprogramming the on-chip flash memory, using the boot loader software and a serial port while the part may reside in the end-user system.
- In Application Programming: In-Application (**IAP**) programming is performing erase and write operation on the on-chip flash memory as directed by the end-user application code.

APPLICATIONS

The flash boot loader provides both In-System and In-Application programming interfaces for programming the on-chip flash memory.

DESCRIPTION

The flash boot loader code is executed every time the part is powered on or reset. The loader can execute the ISP command handler or the user application code. A LOW level after reset at the P0.14 pin is considered as the external hardware request to start the ISP command handler. This pin is sampled in software. Asuming that proper signal is present on X1 pin when the rising edge on RST pin is generated, it may take up to 3 ms before P0.14 is sampled and the decision on wether to continue with user code or ISP handler is made. If P0.14 is sampled low and the watchdog overflow flag is set, the external hardware request to start the ISP command handler is ignored. If there is no request for the ISP command handler execution (P0.14 is sampled HIGH after reset), a search is made for a valid user program. If a valid user program is found then the execution control is transferred to it. If a valid user program is not found, the auto-baud routine is invoked.

Pin P0.14 that is used as hardware request for ISP requires special attention. Since P0.14 is in high impedance mode after reset, it is important that the user provides external hardware (a pull-up resistor or other device) to put the pin in a defined state. Otherwise unintended entry into ISP mode may occur.

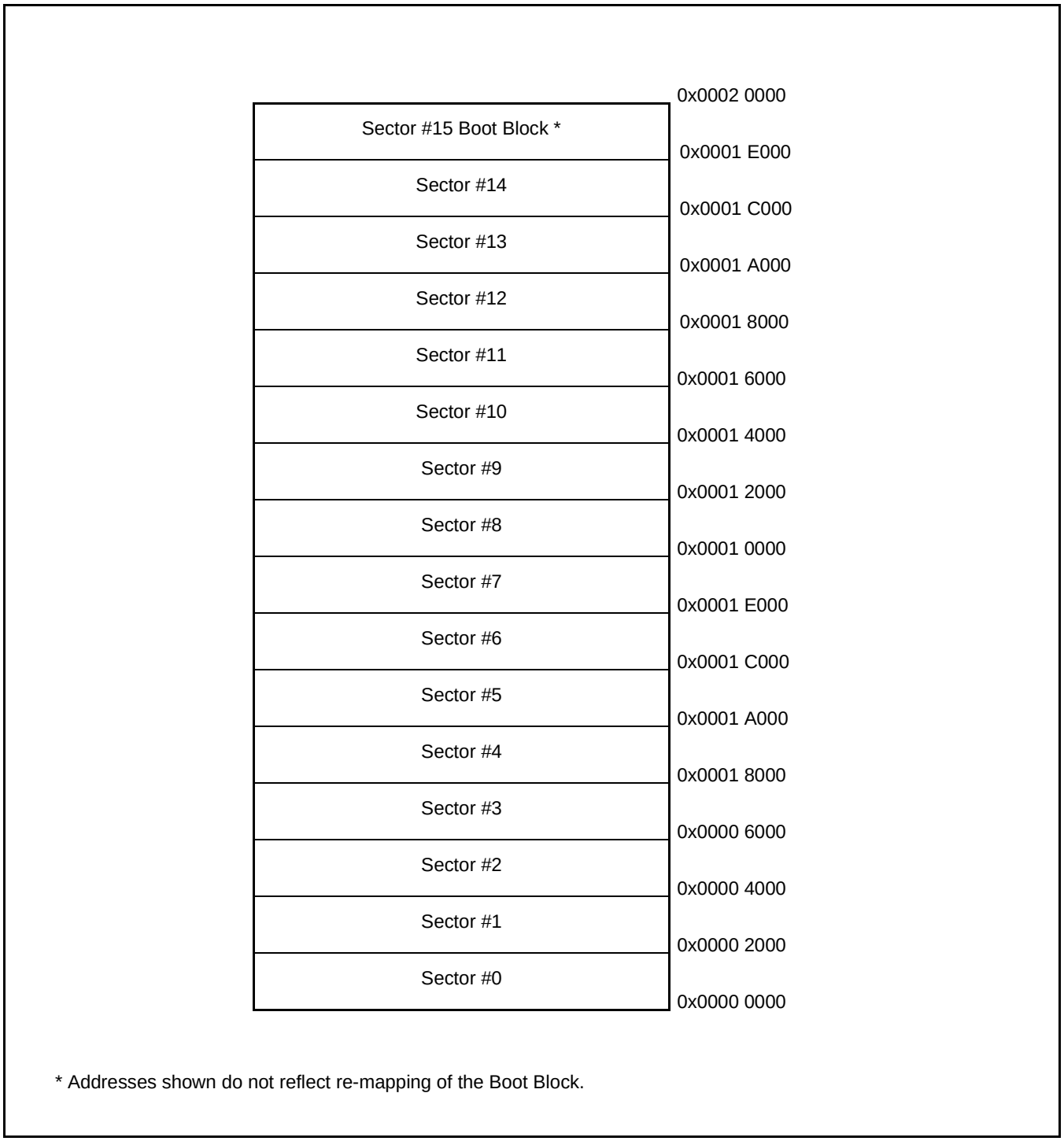


Figure 36: Flash Sector Map

Memory map after any reset:

The boot sector is 8 kB in size and resides in the top portion (starting from 0x0001 E000) of the on-chip flash memory. After any reset the entire boot sector is also mapped to the top of the on-chip memory space i.e. the boot sector is also visible in the memory region starting from the address 0x7FFF E000. The flash boot loader is designed to run from this memory area but both the ISP

and IAP software use parts of the on-chip RAM. The RAM usage is described later in this chapter. The interrupt vectors residing in the boot sector of the on-chip flash memory also become active after reset i.e. the bottom 64 bytes of the boot sector are also visible in the memory region starting from the address 0x0000 0000. The reset vector contains a jump instruction to the entry point of the flash boot loader software.

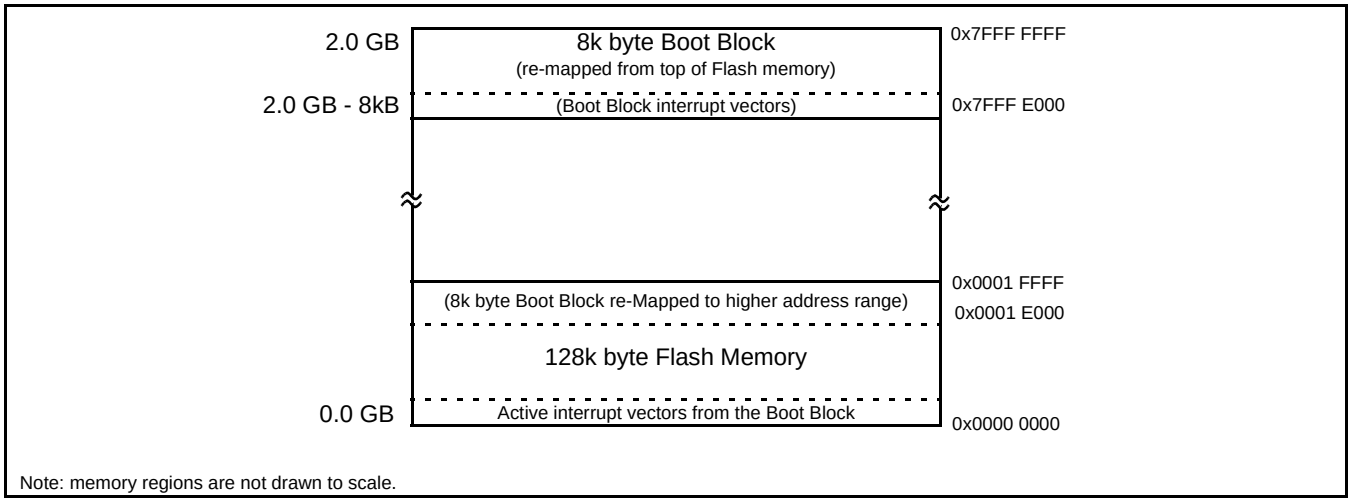


Figure 37: Map of lower memory after any reset.

Criterion for valid user code: The reserved ARM interrupt vector location (0x0000 0014) should contain the 2's complement of the check-sum of the remaining interrupt vectors. This causes the checksum of all of the vectors together to be 0. The boot loader code disables the overlaying of the interrupt vectors from the boot block, then calculates the checksum of the interrupt vectors in sector 0 of the flash. If the signatures match then the execution control is transferred to the user code by loading the program counter with 0x 0000 0000. Hence the user flash reset vector should contain a jump instruction to the entry point of the user application code.

If the signature is not valid, the auto-baud routine synchronizes with the host via serial port 0. The host should send a synchronization character('?') and wait for a response. The host side serial port settings should be 8 data bits, 1 stop bit and no parity. The auto-baud routine measures the bit time of the received synchronization character in terms of its own frequency and programs the baud rate generator of the serial port. It also sends an ASCII string ("Synchronized<CR><LF>") to the host. In response to this the host should send the received string ("Synchronized<CR><LF>"). The auto-baud routine looks at the received characters to verify synchronization. If synchronization is verified then "OK<CR><LF>" string is sent to the host. The host should respond by sending the crystal frequency (in kHz) at which the part is running. For example if the part is running at 10 MHz a valid response from the host should be "10000<CR><LF>". "OK<CR><LF>" string is sent to the host after receiving the crystal frequency. If synchronization is not verified then the auto-baud routine waits again for a synchronization character. For auto-baud to work correctly, the crystal frequency should be greater than or equal to 10 MHz. The on-chip PLL is not used by the boot code.

Once the crystal frequency is received the part is initialized and the ISP command handler is invoked. For safety reasons an "Unlock" command is required before executing commands resulting in flash erase/write operations and the "Go" command. The rest of the commands can be executed without the unlock command. The "Unlock" command is required to be executed once per ISP session. Unlock command is explained in the "ISP Commands" section.

Communication Protocol

All ISP commands should be sent as single ASCII strings. Strings should be terminated with Carriage Return (CR) and/or Line Feed (LF) control characters. Extra <CR> and <LF> characters are ignored. All ISP responses are sent as <CR><LF> terminated ASCII strings. Data is sent and received in UU-encoded format.

ISP Command Format

"Command Parameter_0 Parameter_1 ... Parameter_n<CR><LF>" "Data" (Applicable only in case of Write commands)

ISP Response Format

"Return_Code<CR><LF>Response_0<CR><LF>Response_1<CR><LF> ... Response_n<CR><LF>" "Data" (Applicable in case of Read commands)

ISP Data Format

The data stream is in UU-encode format. The UU-encode algorithm converts 3 bytes of binary data in to 4 bytes of printable ASCII character set. It is more efficient than Hex format, which converts 1 byte of binary data in to 2 bytes of ASCII hex. The sender should send the check-sum after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters(bytes) i.e. it can hold 45 data bytes. The receiver should compare it with the check-sum of the received bytes. If the check-sum matches then the receiver should respond with "OK<CR><LF>" to continue further transmission. If the check-sum does not match the receiver should respond with "RESEND<CR><LF>". In response the sender should retransmit the bytes.

A description of UU-encode is available at *http://www.wotsit.org.*

ISP Flow control

A software XON/XOFF flow control scheme is used to prevent data loss due to buffer overrun. When the data arrives rapidly, the ASCII control character DC3 (stop) is sent to stop the flow of data. Data flow is resumed by sending the ASCII control character DC1 (start). The host should also support the same flow control scheme.

ISP Command Abort

Commands can be aborted by sending the ASCII control character "ESC". This feature is not documented as a command under "ISP Commands" section. Once the escape code is received the ISP command handler waits for a new command.

Interrupts during ISP

Boot block Interrupt vectors located in the boot sector of the flash are active after any reset.

Interrupts during IAP

The on-chip flash memory is not accessible during erase/write operations. When the user application code starts executing the interrupt vectors from the user flash area are active. The user should either disable interrupts, or ensure that user interrupt vectors are active in RAM and that the interrupt handlers reside in RAM, before making a flash erase/write IAP call. The IAP code does not use or disable interrupts.

RAM used by ISP command handler

ISP commands use on-chip RAM from 0x4000 0120 to 0x4000 01FF. The user could use this area, but the contents may be lost upon reset. Flash programming commands use the top 32 bytes of on-chip RAM. The stack is located at RAM top - 32. The maximum stack usage is 256 bytes and it grows downwards.

RAM used by IAP command handler

Flash programming commands use top 32 bytes of on-chip RAM. The maximum stack usage in the user allocated stack space is 128 bytes and it grows downwards.

RAM used by RealMonitor

The RealMonitor uses on-chip RAM from 0x4000 0040 to 0x4000 011F. The user could use this area if RealMonitor based debug is not required. The Flash boot loader does not initialize the stack for the RealMonitor.

BOOT PROCESS FLOWCHART

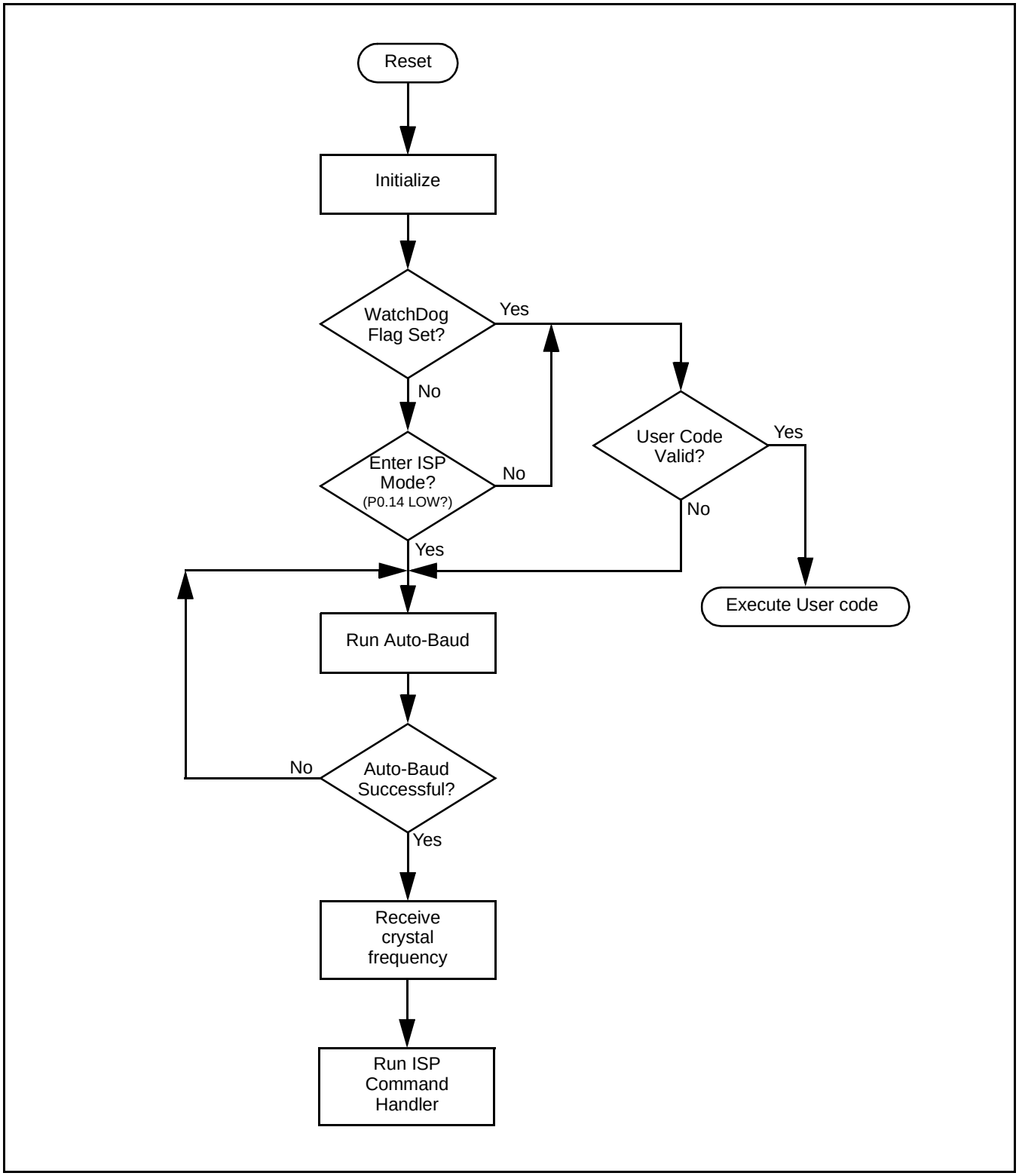


Figure 38: Boot Process flowchart

SECTOR NUMBERS

Some IAP and ISP commands operate on "sectors" and specify sector numbers. The following table indicates the correspondence between sector numbers and memory addresses for LPC2106/2105/2104 device(s). IAP ISP and RealMonitor routines are located in sector 15 (boot sector). The boot sector is present in all devices. ISP and IAP commands do not allow write/erase/go operation on the boot sector. In a device having 128K of Flash, only 120K is available for the user program.

Table 141: Sectors in a device with 128K bytes of Flash

Sector Number	Memory Addresses
0	0x0000 0000 - 1FFF
1	0x0000 2000 - 3FFF
2	0x0000 4000 - 5FFF
3	0x0000 6000 - 7FFF
4	0x0000 8000 - 9FFF
5	0x0000 A000 - BFFF
6	0x0000 C000 - DFFF
7	0x0000 E000 - FFFF
8	0x0001 0000 - 1FFF
9	0x0001 2000 - 3FFF
10 (0x0A)	0x0001 4000 - 5FFF
11 (0x0B)	0x0001 6000 - 7FFF
12 (0x0C)	0x0001 8000 - 9FFF
13 (0x0D)	0x0001 A000 - BFFF
14 (0x0E)	0x0001 C000 - DFFF
15 (0x0F)	0x0001 E000 - FFFF

ISP Commands

The following commands are accepted by the ISP command handler. Detailed return codes are supported for each command. The command handler sends the return code INVALID_COMMAND when an undefined command is received. Commands and return codes are in ASCII format.

CMD_SUCCESS is sent by ISP command handler only when received ISP command has been completely executed and the new ISP command can be given by the host. Exceptions from this rule are "Set Baud Rate", "Write to RAM", "Read Memory", and "Go" commands.

Table 142: ISP Command Summary

ISP Command	Usage	Described in
Unlock	U <Unlock Code>	Table 143
Set Baud Rate	B <Baud Rate> <stop bit>	Table 144
Echo	A <setting>	Table 146
Write to RAM	W <start address> <number of bytes>	Table 147
Read Memory	R <address> <number of bytes>	Table 148
Prepare sector(s) for write operation	P <start sector number> <end sector number>	Table 149
Copy RAM to Flash	C <Flash address> <RAM address> <number of bytes>	Table 150
Go	G <address> <Mode>	Table 151
Erase sector(s)	E <start sector number> <end sector number>	Table 152
Blank check sector(s)	I <start sector number> <end sector number>	Table 153
Read Part ID	J	Table 154
Read Boot code version	K	Table 155
Compare	M <address1> <address2> <number of bytes>	Table 156

Unlock <Unlock code>

Table 143: ISP Unlock command description

Command	U
Input	Unlock code: 23130
Return Code	CMD_SUCCESS | INVALID_CODE | PARAM_ERROR
Description	This command is used to unlock flash Write/Erase & Go commands.
Example	"U 23130<CR><LF>" unlocks the flash Write/Erase & Go commands.

Set Baud Rate <Baud Rate> <stop bit>

Table 144: ISP Set Baud Rate command description

Command	B
Input	Baud Rate: 9600 | 19200 | 38400 | 57600 | 115200 | 230400 Stop bit: 1 | 2
Return Code	CMD_SUCCESS | INVALID_BAUD_RATE | INVALID_STOP_BIT | PARAM_ERROR
Description	This command is used to change the baud rate. The new baud rate is effective after the command handler sends the CMD_SUCCESS return code.
Example	"B 57600 1<CR><LF>" sets the serial port to baud rate 57600 bps and 1 stop bit.

Table 145: Correlation between possible ISP baudrates and external crystal frequency (in MHz)

ISP Baudrate .vs. External Crystal Frequency	9600	19200	38400	57600	115000	230000
10.0000	+	+	+			
11.0592	+	+		+		
12.2880	+	+	+			
14.7456	+	+	+	+	+	+
15.3600	+					
18.4320	+	+		+		
19.6608	+	+	+			
24.5760	+	+	+			
25.0000	+	+	+			
30.0000	+	+	+	+	+	+
44.2350	+	+	+	+	+	+

Echo <setting>

Table 146: ISP Echo command description

Command	A
Input	Setting: ON=1 | OFF=0
Return Code	CMD_SUCCESS | PARAM_ERROR
Description	The default setting for echo command is ON. When ON the ISP command handler sends the received serial data back to the host.
Example	"A 0<CR><LF>" turns echo off.

Write to RAM <start address> <number of bytes>

The host should send the data only after receiving the CMD_SUCCESS return code. The host should send the check-sum after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters(bytes) i.e. it can hold 45 data bytes. When the data fits in less then 20 UU-encoded lines then the check-sum should be of actual number of bytes sent. The ISP command handler compares it with the check-sum of the received bytes. If the check-sum matches then the ISP command handler responds with "OK<CR><LF>" to continue further transmission. If the check-sum does not match then the ISP command handler responds with "RESEND<CR><LF>". In response the host should retransmit the bytes.

Table 147: ISP Write to RAM command description

Command	W
Input	Start Address: RAM address where data bytes are to be written. This address should be a word boundary. Number of Bytes: Number of bytes to be written. Count should be a multiple of 4.
Return Code	CMD_SUCCESS | ADDR_ERROR (Address not a word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not multiple of 4) | PARAM_ERROR
Description	This command is used to download data to RAM. The data should be in UU-encoded format.
Example	"W 1073742336 4<CR><LF>" writes 4 bytes of data to address 0x4000 0200.

Read Memory <address> <number of bytes>

The data stream is followed by the command success return code. The check-sum is sent after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters(bytes) i.e. it can hold 45 data bytes. When the data fits in less then 20 UU-encoded lines then the check-sum is of actual number of bytes sent. The host should compare it with the check-sum of the received bytes. If the check-sum matches then the host should respond with "OK<CR><LF>" to continue further transmission. If the check-sum does not match then the host should respond with "RESEND<CR><LF>". In response the ISP command handler sends the data again.

Table 148: ISP Read Memory command description

Command	R
Input	Start Address: Address from where data bytes are to be read. This address should be a word boundary. Number of Bytes: Number of bytes to be read. Count should be a multiple of 4.
Return Code	CMD_SUCCESS followed by <actual data (UU-encoded)> | ADDR_ERROR (Address not on word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not multiple of 4) | PARAM_ERROR
Description	This command is used to read data from RAM or Flash memory.
Example	"R 1073741824 4<CR><LF>" reads 4 bytes of data from address 0x4000 0000.

Prepare sector(s) for write operation <start sector number> <end sector number>

This command makes flash write/erase operation a two step process.

Table 149: ISP Prepare sector(s) for write operation command description

Command	P
Input	Start Sector Number End Sector Number: Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | BUSY | INVALID_SECTOR | PARAM_ERROR
Description	This command must be executed before executing "Copy RAM to Flash" or "Erase Sector(s)" command. Successful execution of the "Copy RAM to Flash" or "Erase Sector(s)" command causes relevant sectors to be protected again. The boot sector can not be prepared by this command. To prepare a single sector use the same "Start" and "End" sector numbers.
Example	"P 0 0<CR><LF>" prepares the flash sector 0.

Copy RAM to Flash <Flash address> <RAM address> <number of bytes>

Table 150: ISP Copy RAM to Flash command description

Command	C
Input	Flash Address(DST): Destination Flash address where data bytes are to be written. The destination address should be a 512 byte boundary. RAM Address(SRC): Source RAM address from where data bytes are to be read. Number of Bytes: Number of bytes to be written. Should be 512 | 1024 | 4096 | 8192.
Return Code	CMD_SUCCESS | SRC_ADDR_ERROR (Address not on word boundary) | DST_ADDR_ERROR (Address not on correct boundary) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not 512 | 1024 | 4096 | 8192) | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | BUSY | CMD_LOCKED | PARAM_ERROR
Description	This command is used to program the flash memory. The affected sectors should be prepared first by calling "Prepare Sector for Write Operation" command. The affected sectors are automatically protected again once the copy command is successfully executed. The boot sector can not be written by this command.
Example	"C 0 1073774592 512<CR><LF>" copies 512 bytes from the RAM address 0x4000 8000 to the flash address 0.

Go <address> <Mode>

Table 151: ISP Go command description

Command	G
Input	Address: Flash or RAM address from which the code execution is to be started. This address should be on a word boundary. Mode: T (Execute program in Thumb Mode) | A (Execute program in ARM mode)
Return Code	CMD_SUCCESS | ADDR_ERROR | ADDR_NOT_MAPPED | CMD_LOCKED | PARAM_ERROR
Description	This command is used to execute (call) a program residing in RAM or Flash memory. It may not be possible to return to ISP command handler once this command is successfully executed. If executed code has ended with return instruction, ISP handler will resume with execution.
Example	"G 0 A<CR><LF>" branches to address 0x0000 0000 in ARM mode.

Erase sector(s) <start sector number> <end sector number>

Table 152: ISP Erase sector command description

Command	E
Input	Start Sector Number End Sector Number: Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | BUSY | INVALID_SECTOR | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | CMD_LOCKED | PARAM_ERROR
Description	This command is used to erase a sector or multiple sectors of on-chip Flash memory. The boot sector can not be erased by this command. To erase a single sector use the same "Start" and "End" sector numbers.
Example	"E 2 3<CR><LF>" erases the flash sectors 2 and 3.

Blank check sector(s) <start sector number> <end sector number>

Table 153: ISP Blank check sector(s) command description

Command	I
Input	Start Sector Number End Sector Number: Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | SECTOR_NOT_BLANK (followed by <Offset of the first non blank word location> <Contents of non blank word location>) | INVALID_SECTOR | PARAM_ERROR
Description	This command is used to blank check a sector or multiple sectors of on-chip Flash memory. To blank check a single sector use the same "Start" and "End" sector numbers.
Example	"I 2 3<CR><LF>" blank checks the flash sectors 2 and 3. **Blank check on sector 0 always fails as first 64 bytes are re-mapped to flash boot sector.**

Read Part ID

Table 154: ISP Read Part ID command description

Command	J
Input	None
Return Code	CMD_SUCCESS followed by part identification number in ASCII format.
Description	This command is used to read the part identification number.
Example	"J<CR><LF>".

Read Boot code version

Table 155: ISP Read Boot Code version command description

Command	K
Input	None
Return Code	CMD_SUCCESS followed by 2 bytes of boot code version number in ASCII format. It is to be interpreted as <byte1(Major)>.<byte0(Minor)>
Description	This command is used to read the boot code version number.
Example	"K<CR><LF>".

Compare <address1> <address2> <number of bytes>

Table 156: ISP Compare command description

Command	M
Input	Address1(DST): Starting Flash or RAM address from where data bytes are to be compared. This address should be on word boundary. Address2(SRC): Starting Flash or RAM address from where data bytes are to be compared. This address should be on word boundary. Number of Bytes: Number of bytes to be compared. Count should be in multiple of 4.
Return Code	CMD_SUCCESS | (Source and destination data is same) COMPARE_ERROR | (Followed by the offset of first mismatch) COUNT_ERROR (Byte count is not multiple of 4) | ADDR_ERROR | ADDR_NOT_MAPPED | PARAM_ERROR
Description	This command is used to compare the memory contents at two locations.
Example	"M 8192 1073741824 4<CR><LF>" compares 4 bytes from the RAM address 0x4000 0000 to the 4 bytes from the flash address 0x2000. **Compare result may not be correct when source or destination address contains any of the first 64 bytes starting from address zero. First 64 bytes are re-mapped to flash boot sector.**

Table 157: ISP Return Codes Summary

Return Code	Mnemonic	Description
0	CMD_SUCCESS	Command is executed successfully. Sent by ISP handler only when command given by the host has been completely and successfully executed.
1	INVALID_COMMAND	Invalid command.
2	SRC_ADDR_ERROR	Source address is not on word boundary.
3	DST_ADDR_ERROR	Destination address is not on a correct boundary.
4	SRC_ADDR_NOT_MAPPED	Source address is not mapped in the memory map. Count value is taken in to consideration where applicable.
5	DST_ADDR_NOT_MAPPED	Destination address is not mapped in the memory map. Count value is taken in to consideration where applicable.
6	COUNT_ERROR	Byte count is not multiple of 4 or is not a permitted value.
7	INVALID_SECTOR	Sector number is invalid or end sector number is greater than start sector number.
8	SECTOR_NOT_BLANK	Sector is not blank.
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	Command to prepare sector for write operation was not executed.
10	COMPARE_ERROR	Source and destination data not equal.
11	BUSY	Flash programming hardware interface is busy
12	PARAM_ERROR	Insufficient number of parameters or invalid parameter.
13	ADDR_ERROR	Address is not on word boundary.
14	ADDR_NOT_MAPPED	Address is not mapped in the memory map. Count value is taken in to consideration where applicable.
15	CMD_LOCKED	Command is locked.
16	INVALID_CODE	Unlock code is invalid.
17	INVALID_BAUD_RATE	Invalid baud rate setting.
18	INVALID_STOP_BIT	Invalid stop bit setting.

IAP Commands

For in application programming the IAP routine should be called with a word pointer in register r0 pointing to memory (RAM) containing command code and parameters. Result of the IAP command is returned in the result table pointed to by register r1. The user can reuse the command table for result by passing the same pointer in registers r0 and r1. The parameter table should be big enough to hold all the results in case if number of results are more than number of parameters. Parameter passing is illustrated in the Figure 39. The number of parameters and results vary according to the IAP command. The maximum number of parameters is 5, passed to the "Copy RAM to FLASH" command. The maximum number of results is 2, returned by the "Blank check sector(s)" command. The command handler sends the status code INVALID_COMMAND when an undefined command is received. The IAP routine resides at 0x7FFFFFF0 location and it is thumb code.

The IAP function could be called in the following way using C.

Define the IAP location entry point. Since the 0th bit of the IAP location is set there will be a change to Thumb instruction set when the program counter branches to this address.

```
#define IAP_LOCATION 0x7ffffff1
```

Define data structure or pointers to pass IAP command table and result table to the IAP function

```
unsigned long command[5];

unsigned long result[2];
```

or

```
unsigned long * command;

unsigned long * result;

command=(unsigned long *) 0x……

result= (unsigned long *) 0x……
```

Define pointer to function type, which takes two parameters and returns void. Note the IAP returns the result with the base address of the table residing in R1.

```
typedef void (*IAP)(unsigned int [],unsigned int[]);

IAP iap_entry;
```

Setting function pointer

```
iap_entry=(IAP) IAP_LOCATION;
```

Whenever you wish to call IAP you could use the following statement.

```
iap_entry (command, result);
```

The IAP call could be simplified further by using the symbol definition file feature supported by ARM Linker in ADS (ARM Developer Suite). You could also call the IAP routine using assembly code.

The following symbol definitions can be used to link IAP routine and user application:

```
#<SYMDEFS># ARM Linker, ADS1.2 [Build 826]: Last Updated: Wed May 08 16:12:23 2002
0x7fffff90 T rm_init_entry
0x7fffffa0 A rm_undef_handler
```

```
0x7ffffb0 A rm_prefetchabort_handler
0x7ffffc0 A rm_dataabort_handler
0x7ffffd0 A rm_irqhandler
0x7ffffe0 A rm_irqhandler2
0x7fffff0 T iap_entry
```

As per the ARM specification (The ARM Thumb Procedure Call Standard SWS ESPC 0002 A-05) up to 4 parameters can be passed in the r0, r1, r2 and r3 registers respectively. Additional parameters are passed on the stack. Up to 4 parameters can be returned in the r0, r1, r2 and r3 registers respectively. Additional parameters are returned indirectly via memory. Some of the IAP calls require more than 4 parameters. If the ARM suggested scheme is used for the parameter passing/returning then it might create problems due to difference in the C compiler implementation from different vendors. The suggested parameter passing scheme reduces such risk.

The flash memory is not accessible during a write or erase operation. IAP commands, which results in a flash write/erase operation, use 32 bytes of space in the top portion of the on-chip RAM for execution. The user program should not be use this space if IAP flash programming is permitted in the application.

Table 158: IAP Command Summary

IAP Command	Command Code	Described in
Prepare sector(s) for write operation	50	Table 159
Copy RAM to Flash	51	Table 160
Erase sector(s)	52	Table 161
Blank check sector(s)	53	Table 162
Read Part ID	54	Table 163
Read Boot code version	55	Table 164
Compare	56	Table 165

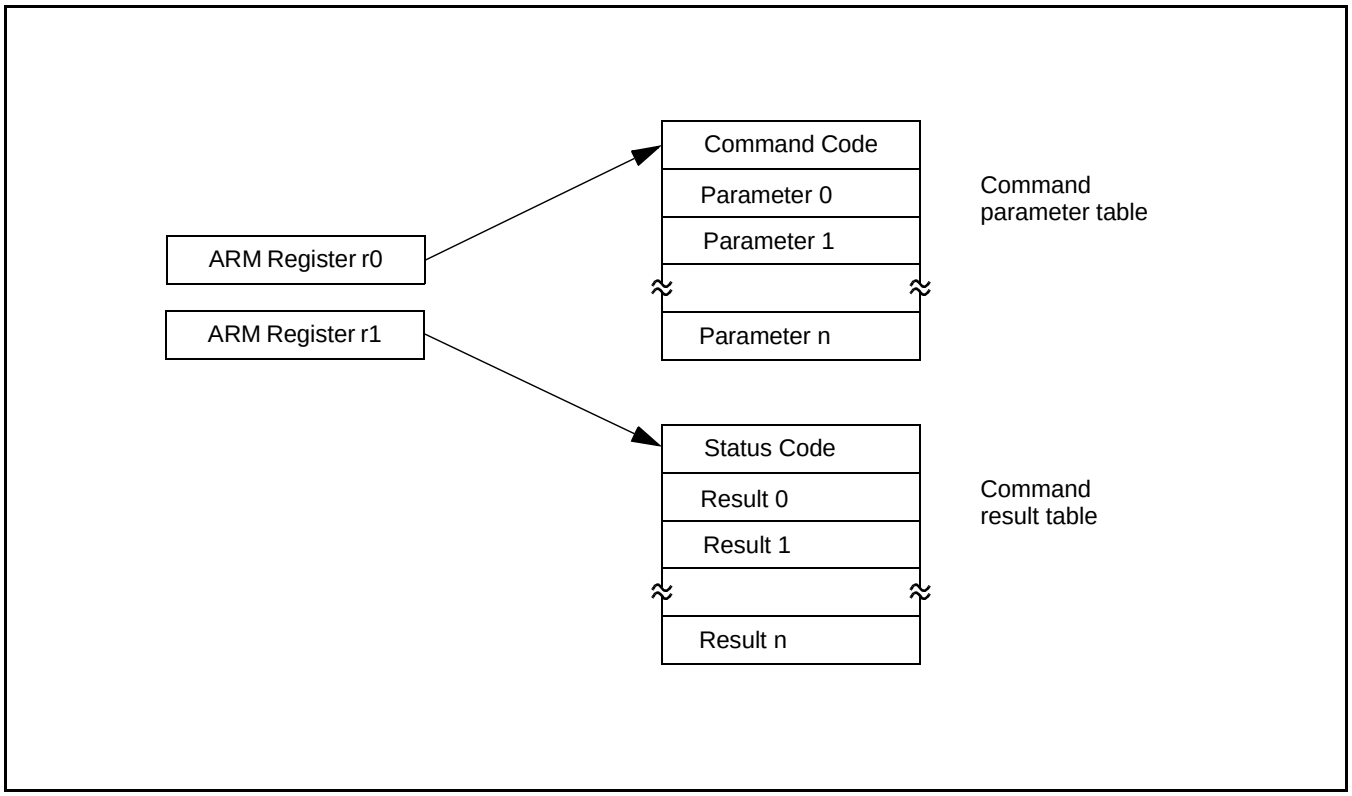


Figure 39: IAP Parameter passing

Prepare sector(s) for write operation

This command makes flash write/erase operation a two step process.

Table 159: IAP Prepare sector(s) for write operation command description

Command	Prepare sector(s) for write operation
Input	Command code: 50 Param0: Start Sector Number Param1: End Sector Number: Should be greater than or equal to start sector number.
Status Code	CMD_SUCCESS | BUSY | INVALID_SECTOR
Result	None
Description	This command must be executed before executing "Copy RAM to Flash" or "Erase Sector(s)" command. Successful execution of the "Copy RAM to Flash" or "Erase Sector(s)" command causes relevant sectors to be protected again. The boot sector can not be prepared by this command. To prepare a single sector use the same "Start" and "End" sector numbers.

Copy RAM to Flash

Table 160: IAP Copy RAM to Flash command description

Command	Copy RAM to Flash
Input	Command code: 51 Param0(DST): Destination Flash address where data bytes are to be written. The destination address should be a 512 byte boundary. Param1(SRC): Source RAM address from which data bytes are to be read. This address should be on word boundary. Param2: Number of bytes to be written. Should be 512 | 1024 | 4096 | 8192. Param3: System Clock Frequency (CCLK) in KHz.
Status Code	CMD_SUCCESS | SRC_ADDR_ERROR (Address not on word boundary) | DST_ADDR_ERROR (Address not on correct boundary) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not 512 | 1024 | 4096 | 8192) | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | BUSY
Result	None
Description	This command is used to program the flash memory. The affected sectors should be prepared first by calling "Prepare Sector for Write Operation" command. The affected sectors are automatically protected again once the copy command is successfully executed. The boot sector can not be written by this command.

Erase Sector(s)

Table 161: IAP Erase Sector(s) command description

Command	Erase Sector(s)
Input	Command code: 52 Param0: Start Sector Number Param1: End Sector Number: Should be greater than or equal to start sector number. Param2: System Clock Frequency (CCLK) in KHz.
Status Code	CMD_SUCCESS | BUSY | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | INVALID_SECTOR
Result	None
Description	This command is used to erase a sector or multiple sectors of on-chip Flash memory. The boot sector can not be erased by this command. To erase a single sector use the same "Start" and "End" sector numbers.

Blank check sector(s)

Table 162: IAP Blank check sector(s) command description

Command	Blank check sector(s)
Input	Command code: 53 Param0: Start Sector Number Param1: End Sector Number: Should be greater than or equal to start sector number.
Status Code	CMD_SUCCESS | BUSY | SECTOR_NOT_BLANK | INVALID_SECTOR
Result	Result0: Offset of the first non blank word location if the Status Code is SECTOR_NOT_BLANK. Result1: Contents of non blank word location.
Description	This command is used to blank check a sector or multiple sectors of on-chip Flash memory. To blank check a single sector use the same "Start" and "End" sector numbers.

Read Part ID

Table 163: IAP Read Part ID command description

Command	Read Part ID
Input	Command Code: 54 parameters: None
Status Code	CMD_SUCCESS
Result	Result0: Part Identification Number
Description	This command is used to read the part identification number.

Read Boot code version

Table 164: IAP Read Boot Code version command description

Command	Read boot code version
Input	Command code: 55 Parameters: None
Status Code	CMD_SUCCESS
Result	Result0: 2 bytes of boot code version number. It is to be interpreted as <byte1(Major)>.<byte0(Minor)>
Description	This command is used to read the boot code version number.

Compare

Table 165: IAP Compare command description

Command	Compare
Input	Command Code: 56 Param0(DST): Starting Flash or RAM address from where data bytes are to be compared. This address should be a word boundary. Param1(SRC): Starting Flash or RAM address from where data bytes are to be compared. This address should be a word boundary. Param2: Number of bytes to be compared. Count should be in multiple of 4.
Status Code	CMD_SUCCESS | COMPARE_ERROR | COUNT_ERROR (Byte count is not multiple of 4) | ADDR_ERROR | ADDR_NOT_MAPPED
Result	Result0: Offset of the first mismatch if the Status Code is COMPARE_ERROR.
Description	This command is used to compare the memory contents at two locations. **Compare result may not be correct when source or destination address contains any of the first 64 bytes starting from address zero. First 64 bytes can be re-mapped to RAM.**

Table 166: IAP Status Codes Summary

Status Code	Mnemonic	Description
0	CMD_SUCCESS	Command is executed successfully.
1	INVALID_COMMAND	Invalid command.
2	SRC_ADDR_ERROR	Source address is not on a word boundary.
3	DST_ADDR_ERROR	Destination address is not on a correct boundary.
4	SRC_ADDR_NOT_MAPPED	Source address is not mapped in the memory map. Count value is taken in to consideration where applicable.
5	DST_ADDR_NOT_MAPPED	Destination address is not mapped in the memory map. Count value is taken in to consideration where applicable.
6	COUNT_ERROR	Byte count is not multiple of 4 or is not a permitted value.
7	INVALID_SECTOR	Sector number is invalid.
8	SECTOR_NOT_BLANK	Sector is not blank.
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	Command to prepare sector for write operation was not executed.
10	COMPARE_ERROR	Source and destination data is not same.
11	BUSY	Flash programming hardware interface is busy.

JTAG FLASH PROGRAMMING INTERFACE

There are three possibilities for this interface:

1. Debug tools can write parts of the flash image to the RAM and then execute the IAP call "Copy RAM to Flash" repeatedly with proper offset.
2. Debug tools can execute the flash programming code via JTAG port.

18. EMBEDDEDICE LOGIC

FEATURES

- No target resources are required by the software debugger in order to start the debugging session
- Allows the software debugger to talk via a JTAG (Joint Test Action Group) port directly to the core
- Inserts instructions directly in to the ARM7TDMI-S core
- The ARM7TDMI-S core or the System state can be examined, saved or changed depending on the type of instruction inserted
- Allows instructions to execute at a slow debug speed or at a fast system speed

APPLICATIONS

The EmbeddedICE logic provides on-chip debug support. The debugging of the target system requires a host computer running the debugger software and an EmbeddedICE protocol convertor. EmbeddedICE protocol convertor converts the Remote Debug Protocol commands to the JTAG data needed to access the ARM7TDMI-S core present on the target system.

DESCRIPTION

The ARM7TDMI-S Debug Architecture uses the existing JTAG* port as a method of accessing the core. The scan chains that are around the core for production test are reused in the debug state to capture information from the databus and to insert new information into the core or the memory. There are two JTAG-style scan chains within the ARM7TDMI-S. A JTAG-style Test Access Port Controller controls the scan chains. In addition to the scan chains, the debug architecture uses EmbeddedICE logic which resides on chip with the ARM7TDMI-S core. The EmbeddedICE has its own scan chain that is used to insert watchpoints and breakpoints for the ARM7TDMI-S core. The EmbeddedICE logic consists of two real time watchpoint registers, together with a control and status register. One or both of the watchpoint registers can be programmed to halt the ARM7TDMI-S core. Execution is halted when a match occurs between the values programmed into the EmbeddedICE logic and the values currently appearing on the address bus, databus and some control signals. Any bit can be masked so that its value does not affect the comparison. Either watchpoint register can be configured as a watchpoint (i.e. on a data access) or a break point (i.e. on an instruction fetch). The watchpoints and breakpoints can be combined such that:

- The conditions on both watchpoints must be satisfied before the ARM7TDMI core is stopped. The CHAIN functionality requires two consecutive conditions to be satisfied before the core is halted. An example of this would be to set the first breakpoint to trigger on an access to a peripheral and the second to trigger on the code segment that performs the task switching. Therefore when the breakpoints trigger the information regarding which task has switched out will be ready for examination.
- The watchpoints can be configured such that a range of addresses are enabled for the watchpoints to be active. The RANGE function allows the breakpoints to be combined such that a breakpoint is to occur if an access occurs in the bottom 256 bytes of memory but not in the bottom 32 bytes.

The ARM7TDMI-S core has a Debug Communication Channel function in-built. The debug communication channel allows a program running on the target to communicate with the host debugger or another separate host without stopping the program flow or even entering the debug state. The debug communication channel is accessed as a co-processor 14 by the program running on the ARM7TDMI-S core. The debug communication channel allows the JTAG port to be used for sending and receiving data without affecting the normal program flow. The debug communication channel data and control registers are mapped in to addresses in the EmbeddedICE logic.

* For more details refer to IEEE Standard 1149.1 - 1990 Standard Test Access Port and Boundary Scan Architecture.

PIN DESCRIPTION

Table 167: EmbeddedICE Pin Description

Pin Name	Type	Description
TMS	Input	**Test Mode Select.** The TMS pin selects the next state in the TAP state machine.
TCK	Input	**Test Clock.** This allows shifting of the data in, on the TMS and TDI pins. It is a positive edge-triggered clock with the TMS and TCK signals that define the internal state of the device.
TDI	Input	**Test Data In.** This is the serial data input for the shift register.
TDO	Output	**Test Data Output.** This is the serial data output from the shift register. Data is shifted out of the device on the negative edge of the TCK signal
nTRST	Input	**Test Reset.** The nTRST pin can be used to reset the test logic within the EmbeddedICE logic.
DBGSEL	Input	**Debug Select.** When low at Reset, the P0.17 - P0.21 pins are configured for alternate functions via the Pin Connect Block. When high at Reset, debug mode is entered. For functionality provided by DBGSEL see "Debug Mode" section of this chapter.
RTCK	Output	**Returned Test Clock.** Extra signal added to the JTAG port. Required for designs based on ARM7TDMI-S processor core. Multi-ICE (Development system from ARM) uses this signal to maintain synchronization with targets having slow or widely varying clock frequency. For details refer to "Multi-ICE System Design considerations Application Note 72 (ARM DAI 0072A)". Also used during entry into debug mode.

REGISTER DESCRIPTION

The EmbeddedICE logic contains 16 registers as shown in Table 168. below. The ARM7TDMI-S debug architecture is described in detail in "ARM7TDMI-S (rev 4) Technical Reference Manual" (ARM DDI 0234A) published by ARM Limited and is available via Internet at http://www.arm.com.

Table 168: EmbeddedICE Logic Registers

Address	Width	Name	Description
00000	6	Debug Control	Force debug state, disable interrupts
00001	5	Debug status	Status of debug
00100	32	Debug Comms Control Register	Debug communication control register
00101	32	Debug Comms Data Register	Debug communication data register
01000	32	Watchpoint 0 Address Value	Holds watchpoint 0 address value
01001	32	Watchpoint 0 Address Mask	Holds watchpoint 0 address mask
01010	32	Watchpoint 0 Data Value	Holds watchpoint 0 data value
01011	32	Watchpoint 0 Data Mask	Holds watchpoint 0 data Mask
01100	9	Watchpoint 0 Control Value	Holds watchpoint 0 control value
01101	8	Watchpoint 0 Control Mask	Holds watchpoint 0 control mask
10000	32	Watchpoint 1 Address Value	Holds watchpoint 1 address value
10001	32	Watchpoint 1 Address Mask	Holds watchpoint 1 address mask
10010	32	Watchpoint 1 Data Value	Holds watchpoint 1 data value
10011	32	Watchpoint 1 Data Mask	Holds watchpoint 1 data Mask
10100	9	Watchpoint 1 Control Value	Holds watchpoint 1 control value
10101	8	Watchpoint 1 Control Mask	Holds watchpoint 1 control mask

BLOCK DIAGRAM

The block diagram of the debug environment is shown below in Figure 40.

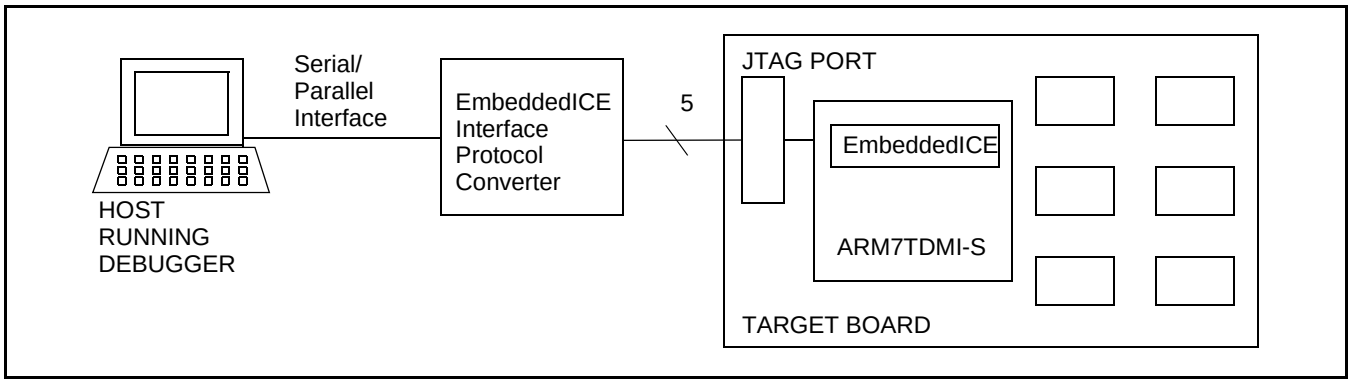


Figure 40: EmbeddedICE Debug Environment Block Diagram

DEBUG MODE

Debug mode connects JTAG pins to embedded ICE for program debugging through the use of an emulator or other development tool.

Enabling Debug Mode

Debug mode is enabled through the use of the DBGSEL and RTCK pins. The two debug variations are selected with the aid of the RTCK pin and software configuration.

To enable the primary debug mode, DBGSEL must be high during and after the CPU is reset. For normal (non-debug) operation, DBGSEL must be kept low at all times (see Figure 41).

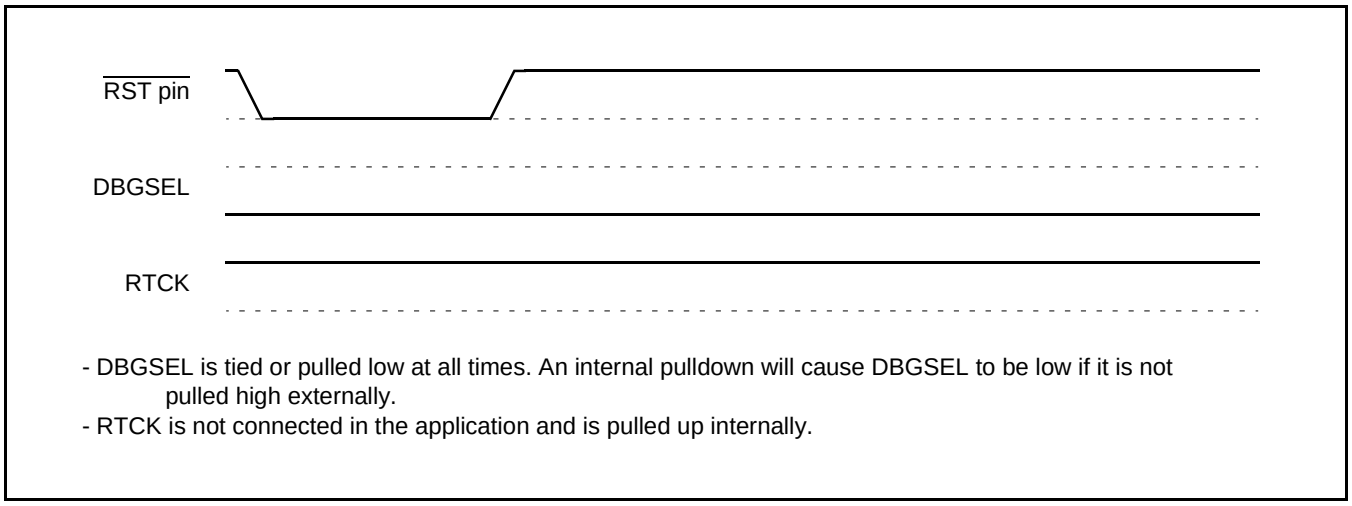


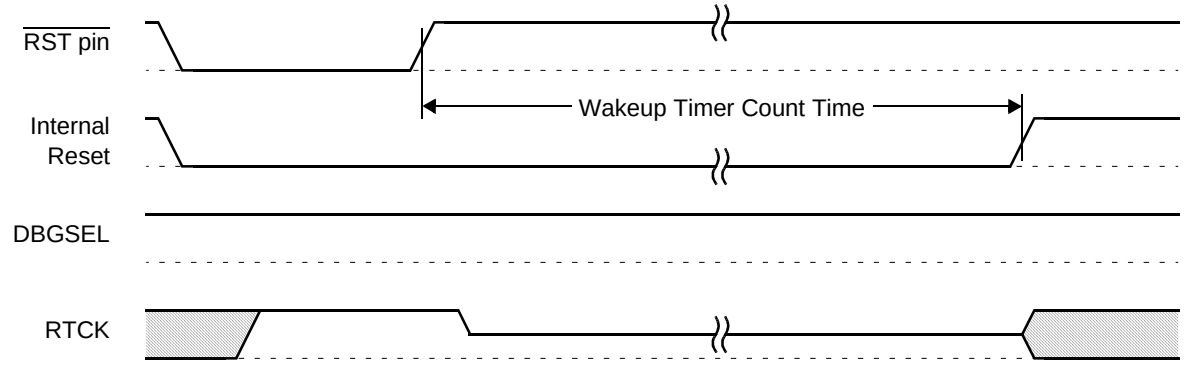
Figure 41: Waveforms for normal operation (not in debug mode)

For debugging with the primary JTAG pins, RTCK must be high as the $\overline{RST}$ pin is released (see Figure 42). RTCK may be driven high externally, or allowed to float high via its on-chip pullup. The RTCK output driver is disabled until the internal wakeup time has expired, allowing an interval between the release of the external reset and the release of the internal reset during which RTCK may be driven by an external signal if necessary.

This procedure establishes the P0.17 - P0.21 pins as the JTAG Test/Debug interface. Pin connect block settings have no affect on P0.17 - P0.21 pins if they are initialized as JTAG pins.

For debugging using the secondary JTAG pins, software must configure the related pins to connect the JTAG port. This is done via the Pin Connect Block.

For effect of hardware override related to DBGSEL and RTCK see Table 45 in "Pin Configuration" chapter. Table 169 shows how JTAG port is selected based on DBGSEL, RTCLK, and user's code executed after the reset.



- DBGSEL has to be high.
- RTCK must be high as RST is released. An internal pullup will cause RTCK to be high if it is not pulled low externally.
- The RTCK output driver will be turned on when the internal chip reset is released by the wakeup timer.

Figure 42: Waveforms for Debug mode using the primary JTAG pins.

Table 169: JTAG Pins Selection

DBGSEL (After Reset)	Latched RTCK Value	JTAG Primary Pins	JTAG Secondaty Pins	ETM
1	1	Yes	No	Yes
0	1	No	SW config*	No
1	0	No	SW config*	No
0	0	No	SW config*	No

*Start-up-code residing in Flash should configure port pins P0.27-P0.31 for JTAG function by setting appropriate bits in PINSEL1 register.

Primary JTAG port and ETM can be selected for debugging only when DBGSEL and RTCK pins are high at reset (see Figure 42). If at least one of DBGSEL or RTCK lines is low at reset, neither primary JTAG nor ETM will be enabled, and they could not be used for later debugging. However, in these cases user's application can assign secondary JTAG functions to pins P0.27 to P0.31. While user's code can redirect JTAG communication to the secondary JTAG pins, ETM functionality will not be available, since ETM and secondary JTAG interface are sharing the same pins and consequently are excluding one another.

19. EMBEDDED TRACE MACROCELL

FEATURES

- Closely track the instructions that the ARM core is executing
- 10 pin interface
- 1 External trigger input
- All registers are programmed through JTAG interface
- Does not consume power when trace is not being used
- THUMB instruction set support

APPLICATIONS

As the microcontroller has significant amounts of on-chip memories, it is not possible to determine how the processor core is operating simply by observing the external pins. The ETM provides real-time trace capability for deeply embedded processor cores. It outputs information about processor execution to a trace port. A software debugger allows configuration of the ETM using a JTAG interface and displays the trace information that has been captured, in a format that a user can easily understand.

DESCRIPTION

The ETM is connected directly to the ARM core and not to the main AMBA system bus. It compresses the trace information and exports it through a narrow trace port. An external Trace Port Analyzer captures the trace information under software debugger control. Trace port can broadcast the Instruction trace information. Instruction trace (or PC trace) shows the flow of execution of the processor and provides a list of all the instructions that were executed. Instruction trace is significantly compressed by only broadcasting branch addresses as well as a set of status signals that indicate the pipeline status on a cycle by cycle basis. Trace information generation can be controlled by selecting the trigger resource. Trigger resources include address comparators, counters and sequencers. Since trace information is compressed the software debugger requires a static image of the code being executed. Self-modifying code can not be traced because of this restriction.

ETM Configuration

The following standard configuration is selected for the ETM macrocell.

Table 170: ETM Configuration

Resource number/type	Small [1]
Pairs of address comparators	1
Data Comparators	0 (Data tracing is not supported)
Memory Map Decoders	4
Counters	1
Sequencer Present	No
External Inputs	2

Table 170: ETM Configuration

Resource number/type	Small [1]
External Outputs	0
FIFOFULL Present	Yes (Not wired)
FIFO depth	10 bytes
Trace Packet Width	4/8

1. For details refer to ARM documentation "Embedded Trace Macrocell Specification (ARM IHI 0014E)".

PIN DESCRIPTION

Table 171: ETM Pin Description

Pin Name	Type	Description
TRACECLK	Output	**Trace Clock.** The trace clock signal provides the clock for the trace port. PIPESTAT[2:0], TRACESYNC, and TRACEPKT[3:0] signals are referenced to the rising edge of the trace clock. This clock is not generated by the ETM block. It is to be derived from the system clock. The clock should be balanced to provide sufficient hold time for the trace data signals. Half rate clocking mode is supported. Trace data signals should be shifted by a clock phase from TRACECLK. Refer to Figure 3.14 page 3.26 and figure 3.15 page 3.27 in "ETM7 Technical Reference Manual" (ARM DDI 0158B), for example circuits that implements both half-rate-clocking and shifting of the trace data with respect to the clock. For TRACECLK timings refer to section 5.2 on page 5-13 in "Embedded Trace Macrocell Specification" (ARM IHI 0014E).
PIPESTAT[2:0]	Output	**Pipe Line status.** The pipeline status signals provide a cycle-by-cycle indication of what is happening in the execution stage of the processor pipeline.
TRACESYNC	Output	**Trace synchronization.** The trace sync signal is used to indicate the first packet of a group of trace packets and is asserted HIGH only for the first packet of any branch address.
TRACEPKT[3:0]	Output	**Trace Packet.** The trace packet signals are used to output packaged address and data information related to the pipeline status. All packets are eight bits in length. A packet is output over two cycles. In the first cycle, Packet[3:0] is output and in the second cycle, Packet[7:4] is output.
EXTIN[0]	Input	External Trigger Input.

REGISTER DESCRIPTION

The ETM contains 29 registers as shown in Table 172. below. They are described in detail in the ARM IHI 0014E document published by ARM Limited, which is available via the Internet at http://www.arm.com.

Table 172: ETM Registers

Register encoding	Name	Description	Access
000 0000	ETM Control	Controls the general operation of the ETM	Read/Write
000 0001	ETM Configuration Code	Allows a debugger to read the number of each type of resource	Read Only
000 0010	Trigger Event	Holds the controlling event	Write Only
000 0011	Memory Map Decode Control	Eight-bit register, used to statically configure the memory map decoder	Write Only
000 0100	ETM Status	Holds the pending overflow status bit	Read Only
000 0101	System Configuration	Holds the configuration information using the SYSOPT bus	Read Only
000 0110	Trace Enable Control 3	Holds the trace on/off addresses	Write Only
000 0111	Trace Enable Control 2	Holds the address of the comparison	Write Only
000 1000	Trace Enable Event	Holds the enabling event	Write Only
000 1001	Trace Enable Control 1	Holds the include and exclude regions	Write Only
000 1010	FIFOFULL Region	Holds the include and exclude regions	Write Only
000 1011	FIFOFULL Level	Holds the level below which the FIFO is considered full	Write Only
000 1100	ViewData event	Holds the enabling event	Write Only
000 1101	ViewData Control 1	Holds the include/exclude regions	Write Only
000 1110	ViewData Control 2	Holds the include/exclude regions	Write Only
000 1111	ViewData Control 3	Holds the include/exclude regions	Write Only
001 xxxx	Address Comparator 1 to 16	Holds the address of the comparison	Write Only
010 xxxx	Address Access Type 1 to 16	Holds the type of access and the size	Write Only
000 xxxx	reserved	-	-
100 xxxx	reserved	-	-
101 00xx	Initial Counter Value 1 to 4	Holds the initial value of the counter	Write Only
101 01xx	Counter Enable 1 to 4	Holds the counter clock enable control and event	Write Only
101 10xx	Counter reload 1 to 4	Holds the counter reload event	Write Only
101 11xx	Counter Value 1 to 4	Holds the current counter value	Read Only
110 00xx	Sequencer State and Control	Holds the next state triggering events.	-
110 10xx	External Output 1 to 4	Holds the controlling events for each output	Write Only
110 11xx	Reserved	-	-
111 0xxx	Reserved	-	-
111 1xxx	Reserved	-	-

BLOCK DIAGRAM

The block diagram of the ETM debug environment is shown below in Figure 43.

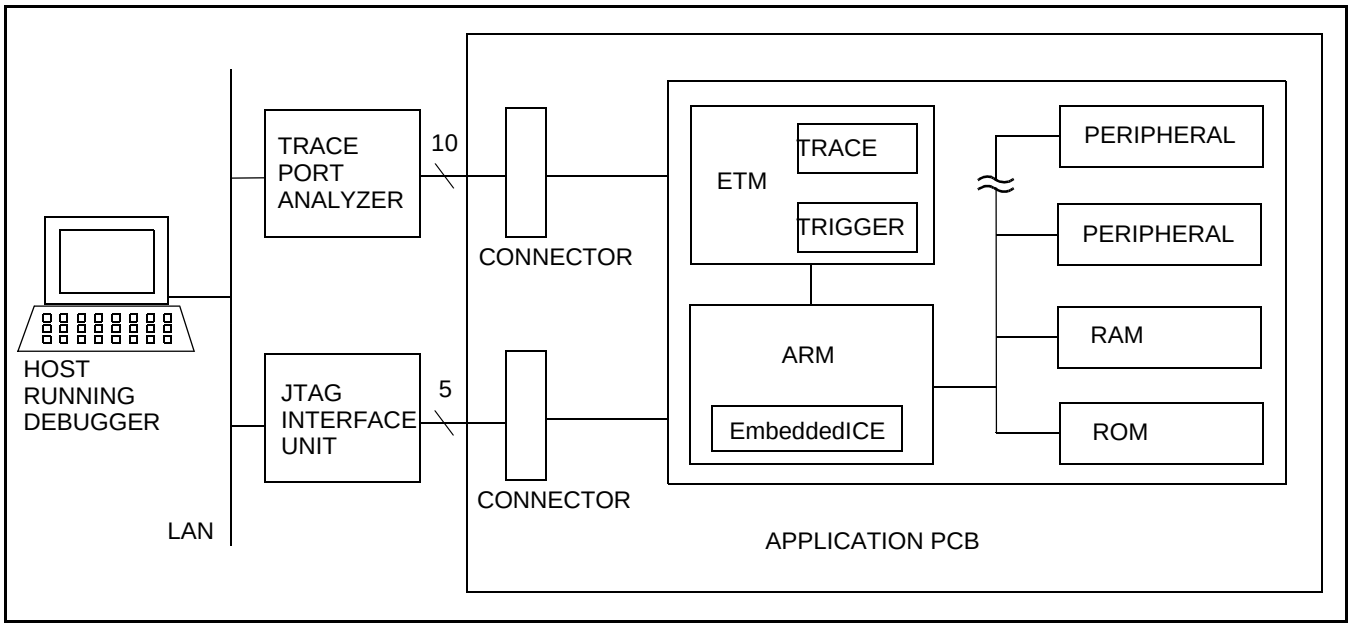


Figure 43: ETM Debug Environment Block Diagram

20. REALMONITOR

RealMonitor is a configurable software module which enables real time debug. RealMonitor is developed by ARM Inc. Information presented in this chapter is taken from the ARM document *RealMonitor Target Integration Guide (ARM DUI 0142A).* It applies to a specific configuration of RealMonitor software programmed in the on-chip flash memory of this device.

Refer to the white paper *"Real Time Debug for System-on-Chip"* available at *http://www.arm.com/support/White_Papers?OpenDocument* for background information.

FEATURES

- Allows user to establish a debug session to a currently running system without halting or resetting the system.
- Allows user time-critical interrupt code to continue executing while other user application code is being debugged.

APPLICATIONS

Real time debugging.

DESCRIPTION

RealMonitor is a lightweight debug monitor that allows interrupts to be serviced while user debug their foreground application. It communicates with the host using the DCC (Debug Communications Channel), which is present in the EmbeddedICE logic. RealMonitor provides advantages over the traditional methods for debugging applications in ARM systems. The traditional methods include:

- Angel (a target-based debug monitor).
- Multi-ICE or other JTAG unit and EmbeddedICE logic (a hardware-based debug solution).

Although both of these methods provide robust debugging environments, neither is suitable as a lightweight real-time monitor.

Angel is designed to load and debug independent applications that can run in a variety of modes, and communicate with the debug host using a variety of connections (such as a serial port or ethernet). Angel is required to save and restore full processor context, and the occurrence of interrupts can be delayed as a result. Angel, as a fully functional target-based debugger, is therefore too heavyweight to perform as a real-time monitor.

Multi-ICE is a hardware debug solution that operates using the EmbeddedICE unit that is built into most ARM processors. To perform debug tasks such as accessing memory or the processor registers, Multi-ICE must place the core into a debug state. While the processor is in this state, which can be millions of cycles, normal program execution is suspended, and interrupts cannot be serviced.

RealMonitor combines features and mechanisms from both Angel and Multi-ICE to provide the services and functions that are required. In particular, it contains both the Multi-ICE communication mechanisms (the DCC using JTAG), and Angel-like support for processor context saving and restoring. RealMonitor is pre-programmed in the on-chip Flash memory (boot sector). When enabled It allows user to observe and debug while parts of application continue to run. Refer to section How to Enable RealMonitor for details.

RealMonitor Components

As shown in Figure 44, RealMonitor is split in to two functional components:

RMHost

This is located between a debugger and a JTAG unit. The RMHost controller, RealMonitor.dll, converts generic *Remote Debug Interface* (RDI) requests from the debugger into DCC-only RDI messages for the JTAG unit. For complete details on debugging a RealMonitor-integrated application from the host, see the *ARM RMHost User Guide (ARM DUI 0137A).*

RMTarget

This is pre-programmed in the on-chip Flash memory (boot sector), and runs on the target hardware. It uses the EmbeddedICE logic, and communicates with the host using the DCC. For more details on RMTarget functionality, see the *RealMonitor Target Integration Guide (ARM DUI 0142A).*

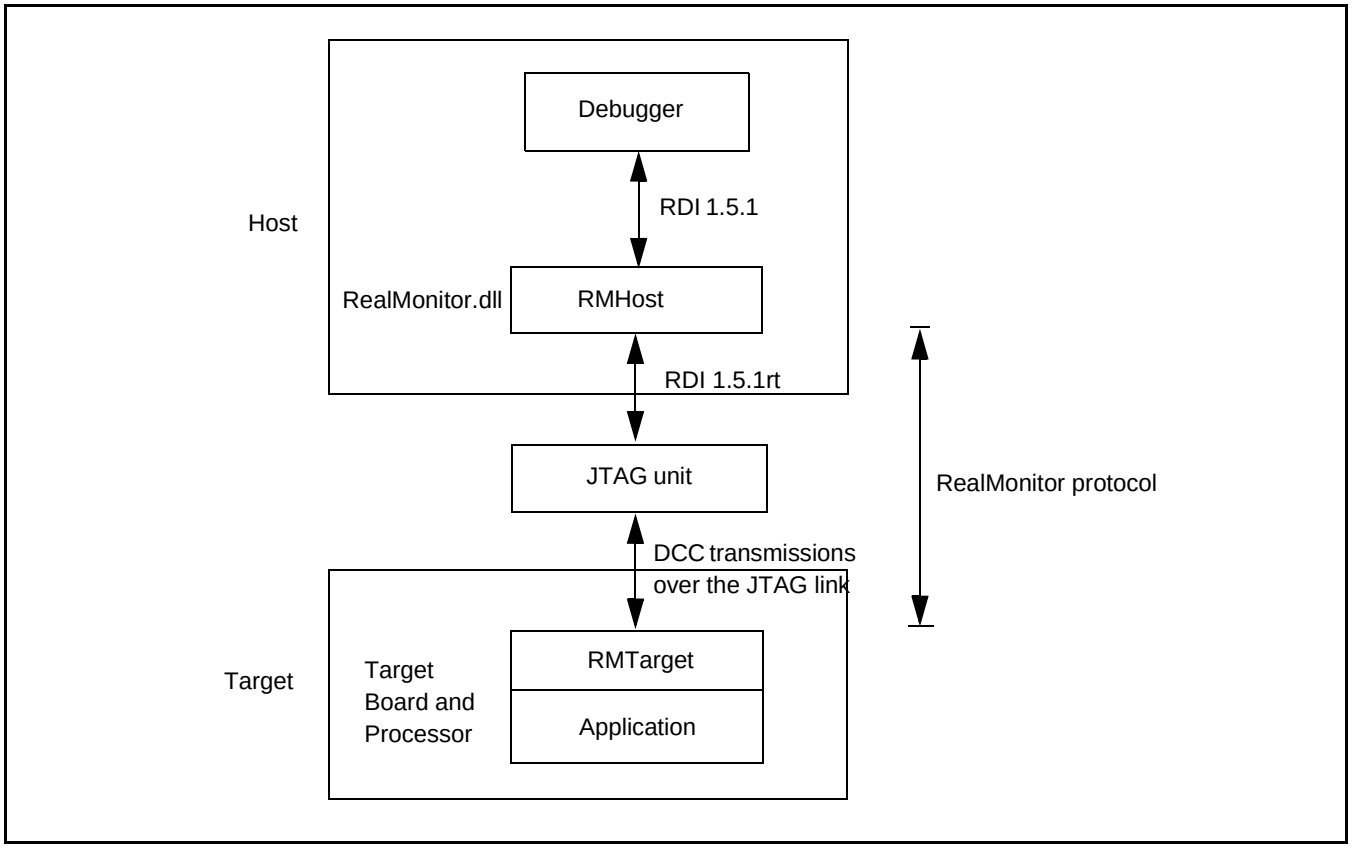


Figure 44: RealMonitor components

How RealMonitor works

In general terms, the RealMonitor operates as a state machine, as shown in Figure 45. RealMonitor switches between running and stopped states, in response to packets received by the host, or due to asynchronous events on the target. RMTarget supports the triggering of only one breakpoint, watchpoint, stop, or semihosting SWI at a time. There is no provision to allow nested events to be saved and restored. So, for example, if user application has stopped at one breakpoint, and another breakpoint occurs in an IRQ handler, RealMonitor enters a panic state. No debugging can be performed after RealMonitor enters this state.

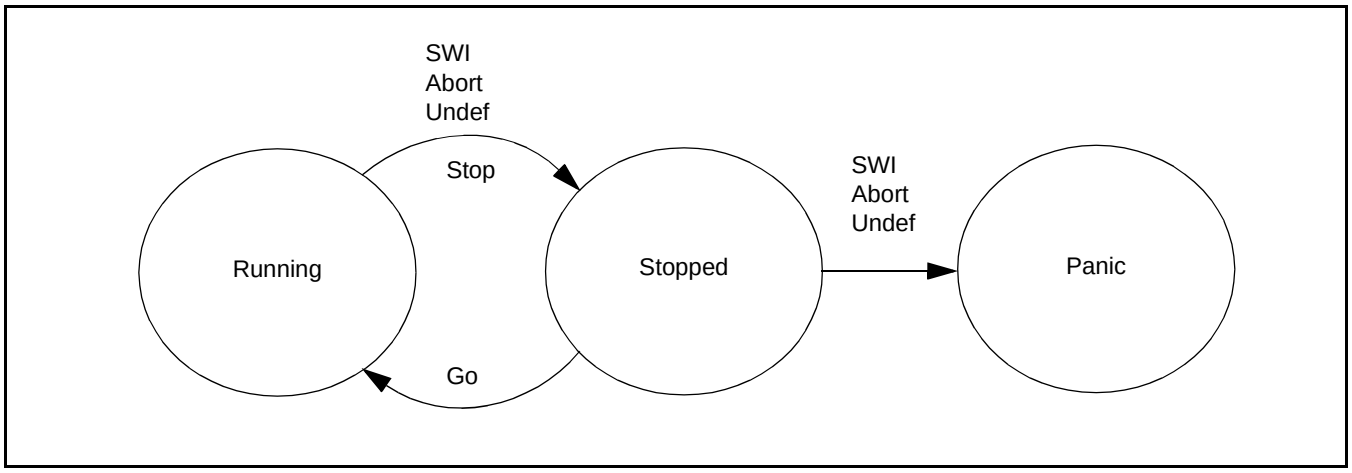


Figure 45: RealMonitor as a state machine

A debugger such as the ARM eXtended Debugger (AXD) or other RealMonitor aware debugger, that runs on a host computer, can connect to the target to send commands and receive data. This communication between host and target is illustrated in Figure 44.

The target component of RealMonitor, RMTarget, communicates with the host component, RMHost, using the Debug Communications Channel (DCC), which is a reliable link whose data is carried over the JTAG connection.

While user application is running, RMTarget typically uses IRQs generated by the DCC. This means that if user application also wants to use IRQs, it must pass any DCC-generated interrupts to RealMonitor.

To allow nonstop debugging, the EmbeddedICE-RT logic in the processor generates a Prefetch Abort exception when a breakpoint is reached, or a Data Abort exception when a watchpoint is hit. These exceptions are handled by the RealMonitor exception handlers that inform the user, by way of the debugger, of the event. This allows user application to continue running without stopping the processor. RealMonitor considers user application to consist of two parts:

- a foreground application running continuously, typically in User, System, or SVC mode
- a background application containing interrupt and exception handlers that are triggered by certain events in user system, including:
  - IRQs or FIQs
  - Data and Prefetch aborts caused by user foreground application. This indicates an error in the application being debugged. In both cases the host is notified and the user application is stopped.
  - Undef exception caused by the undefined instructions in user foreground application. This indicates an error in the application being debugged. RealMonitor stops the user application until a "Go" packet is received from the host.

When one of these exceptions occur that is not handled by user application, the following happens:

- RealMonitor enters a loop, polling the DCC. If the DCC read buffer is full, control is passed to rm_ReceiveData() (RealMonitor internal function). If the DCC write buffer is free, control is passed to rm_TransmitData() (RealMonitor internal function). If there is nothing else to do, the function returns to the caller. The ordering of the above comparisons gives reads from the DCC a higher priority than writes to the communications link.
- RealMonitor stops the foreground application. Both IRQs and FIQs continue to be serviced if they were enabled by the application at the time the foreground application was stopped.

HOW TO ENABLE REALMONITOR

The following steps must be performed to enable RealMonitor. A code example which implements all the steps can be found at the end of this section.

Adding stacks

User must ensure that stacks are set up within application for each of the processor modes used by RealMonitor. For each mode, RealMonitor requires a fixed number of words of stack space. User must therefore allow sufficient stack space for both RealMonitor and application.

RealMonitor has the following stack requirements:

Table 173: RealMonitor stack requirement

Processor Mode	RealMonitor Stack Usage (Bytes)
Undef	48
Prefetch Abort	16
Data Abort	16
IRQ	8

IRQ mode

A stack for this mode is always required. RealMonitor uses two words on entry to its interrupt handler. These are freed before nested interrupts are enabled.

Undef mode

A stack for this mode is always required. RealMonitor uses 12 words while processing an undefined instruction exception.

SVC mode

RealMonitor makes no use of this stack.

Prefetch Abort mode

RealMonitor uses four words on entry to its Prefetch abort interrupt handler.

Data Abort mode

RealMonitor uses four words on entry to its data abort interrupt handler.

User/System mode

RealMonitor makes no use of this stack.

FIQ mode

RealMonitor makes no use of this stack.

Handling exceptions

This section describes the importance of sharing exception handlers between RealMonitor and user application.

RealMonitor exception handling

To function properly, RealMonitor must be able to intercept certain interrupts and exceptions. Figure 46 illustrates how exceptions can be claimed by RealMonitor itself, or shared between RealMonitor and application. If user application requires the exception sharing, they must provide function (such as *app_IRQDispatch* ()). Depending on the nature of the exception, this handler can either:

- pass control to the RealMonitor processing routine, such as rm_irqhandler2()
- claim the exception for the application itself, such as *app_IRQHandler* ().

In a simple case where an application has no exception handlers of its own, the application can install the RealMonitor low-level exception handlers directly into the vector table of the processor. Although the irq handler must get the address of the Vectored Interrupt Controller. The easiest way to do this is to write a branch instruction (<*address* >) into the vector table, where the target of the branch is the start address of the relevant RealMonitor exception handler.

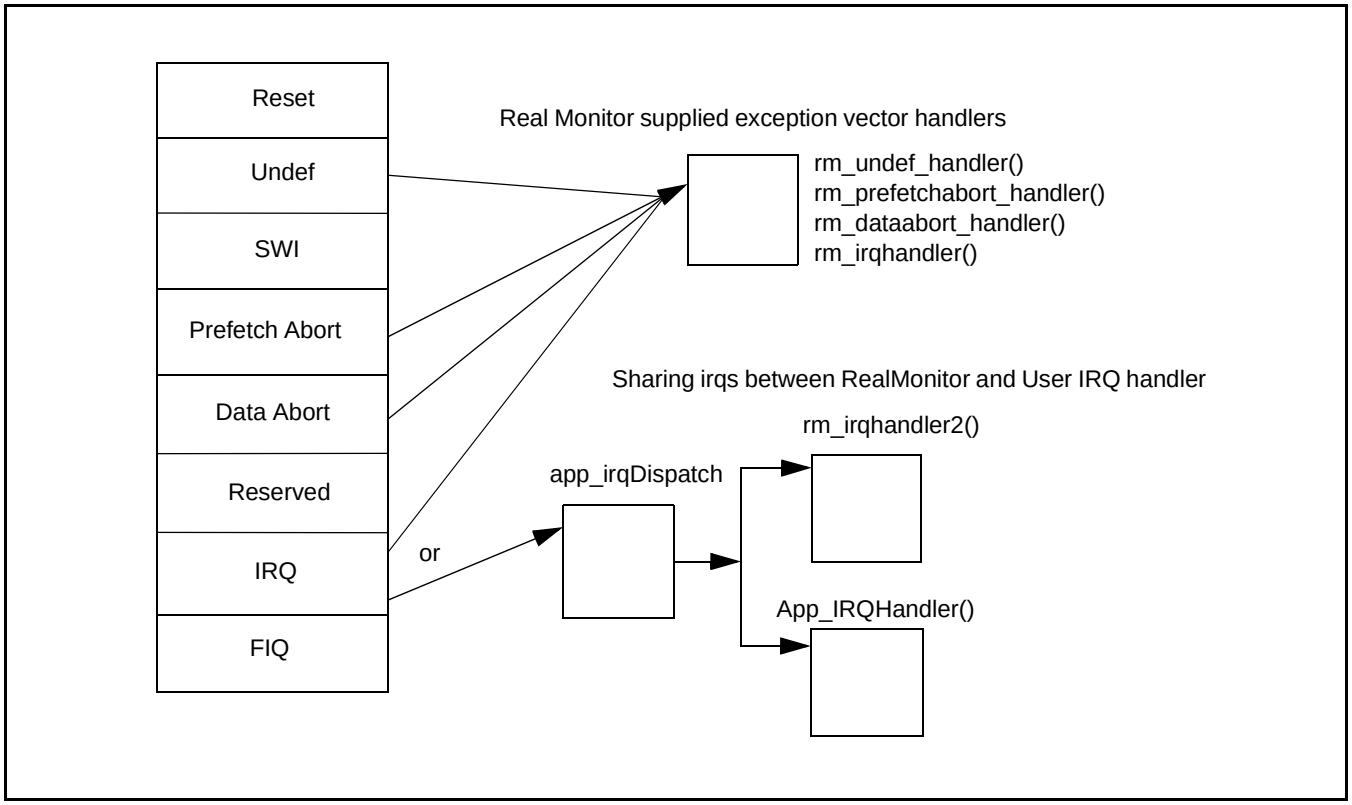


Figure 46: Exception Handlers

RMTarget initialization

While the processor is in a privileged mode, and IRQs are disabled, user must include a line of code within the start-up sequence of application to call rm_init_entry().

Code Example

The following example shows how to setup stack, VIC, initialize RealMonitor and share non vectored interrupts:

```
    IMPORT rm_init_entry
    IMPORT rm_prefetchabort_handler
    IMPORT rm_dataabort_handler
    IMPORT rm_irqhandler2
    IMPORT rm_undef_handler
    IMPORT User_Entry ;Entry point of user application.
    CODE32
    ENTRY
   ;Define exception table. Instruct linker to place code at address 0x0000 0000

    AREA exception_table, CODE

    LDR pc, Reset_Address
    LDR pc, Undefined_Address
    LDR pc, SWI_Address
    LDR pc, Prefetch_Address
    LDR pc, Abort_Address
    NOP ; Insert User code valid signature here.
    LDR pc, [pc, #-0xFF0] ;Load IRQ vector from VIC
    LDR PC, FIQ_Address

Reset_Address      DCD     __init                   ;Reset Entry point
Undefined_Address  DCD     rm_undef_handler         ;Provided by RealMonitor
SWI_Address        DCD     0                        ;User can put address of SWI handler here
Prefetch_Address   DCD     rm_prefetchabort_handler ;Provided by RealMonitor
Abort_Address      DCD     rm_dataabort_handler     ;Provided by RealMonitor
FIQ_Address        DCD     0                        ;User can put address of FIQ handler here

AREA init_code, CODE

ram_end EQU 0x4000xxxx ; Top of on-chip RAM.
__init
; /*********************************************************************
; * Set up the stack pointers for various processor modes. Stack grows
; * downwards.
; *********************************************************************/
    LDR r2, =ram_end    ;Get top of RAM
    MRS r0, CPSR        ;Save current processor mode
    ; Initialize the Undef mode stack for RealMonitor use
    BIC     r1, r0, #0x1f
    ORR     r1, r1, #0x1b
    MSR     CPSR_c, r1
    ;Keep top 32 bytes for flash programming routines.
    ;Refer to Flash Memory System and Programming chapter
    SUB sp,r2,#0x1F

    ; Initialize the Abort mode stack for RealMonitor
    BIC     r1, r0, #0x1f
    ORR     r1, r1, #0x17
    MSR     CPSR_c, r1
    ;Keep 64 bytes for Undef mode stack
    SUB sp,r2,#0x5F
```

```
    ; Initialize the IRQ mode stack for RealMonitor and User
    BIC     r1, r0, #0x1f
    ORR     r1, r1, #0x12
    MSR     CPSR_c, r1
    ;Keep 32 bytes for Abort mode stack
    SUB sp,r2,#0x7F

    ; Return to the original mode.
    MSR     CPSR_c, r0

    ; Initialize the stack for user application
    ; Keep 256 bytes for IRQ mode stack
    SUB sp,r2,#0x17F

; /*********************************************************************
; * Setup Vectored Interrupt controller. DCC Rx and Tx interrupts
; * generate Non Vectored IRQ request. rm_init_entry is aware
; * of the VIC and it enables the DBGCommRX and DBGCommTx interrupts.
; * Default vector address register is programmed with the address of
; * Non vectored app_irqDispatch mentioned in this example. User can setup
; * Vectored IRQs or FIQs here.
; *********************************************************************/

    VICBaseAddr          EQU 0xFFFFF000    ; VIC Base address
    VICDefVectAddrOffset EQU 0x34

    LDR r0, =VICBaseAddr
    LDR r1, =app_irqDispatch
    STR r1, [r0,#VICDefVectAddrOffset]

    BL rm_init_entry ;Initialize RealMonitor
    ;enable FIQ and IRQ in ARM Processor
    MRS     r1, CPSR        ; get the CPSR
    BIC     r1, r1, #0xC0   ; enable IRQs and FIQs
    MSR     CPSR_c, r1      ; update the CPSR
; /*********************************************************************
; * Get the address of the User entry point.
; *********************************************************************/
    LDR lr, =User_Entry
    MOV pc, lr
; /*********************************************************************
; * Non vectored irq handler (app_irqDispatch)
; *********************************************************************/

AREA app_irqDispatch, CODE
VICVectAddrOffset EQU 0x30
app_irqDispatch

    ;enable interrupt nesting
    STMFD sp!, {r12,r14}
    MRS r12, spsr             ;Save SPSR in to r12
    MSR cpsr_c,0x1F           ;Re-enable IRQ, go to system mode

    ;User should insert code here if non vectored Interrupt sharing is
    ;required. Each non vectored shared irq handler must return to
    ;the interrupted instruction by using the following code.
    ;MSR cpsr_c, #0x52              ;Disable irq, move to IRQ mode
```

```
;MSR spsr, r12                  ;Restore SPSR from r12
;STMFD sp!, {r0}
;LDR r0, =VICBaseAddr
;STR r1, [r0,#VICVectAddrOffset] ;Acknowledge Non Vectored irq has finished
;LDMFD sp!, {r12,r14,r0}         ;Restore registers
;SUBS pc, r14, #4                ;Return to the interrupted instruction

;user interrupt did not happen so call rm_irqhandler2. This handler
;is not aware of the VIC interrupt priority hardware so trick
;rm_irqhandler2 to return here

STMFD sp!, {ip,pc}
LDR pc, rm_irqhandler2
;rm_irqhandler2 returns here
MSR cpsr_c, #0x52               ;Disable irq, move to IRQ mode
MSR spsr, r12                   ;Restore SPSR from r12
STMFD sp!, {r0}
LDR r0, =VICBaseAddr
STR r1, [r0,#VICVectAddrOffset] ;Acknowledge Non Vectored irq has finished
LDMFD sp!, {r12,r14,r0}         ;Restore registers
SUBS pc, r14, #4                ;Return to the interrupted instruction

END
```

REALMONITOR BUILD OPTIONS

RealMonitor was built with the following options:

RM_OPT_DATALOGGING=FALSE

This option enables or disables support for any target-to-host packets sent on a non RealMonitor (third-party) channel.

RM_OPT_STOPSTART=TRUE

This option enables or disables support for all stop and start debugging features.

RM_OPT_SOFTBREAKPOINT=TRUE

This option enables or disables support for software breakpoints.

RM_OPT_HARDBREAKPOINT=TRUE

Enabled for cores with EmbeddedICE-RT. This device uses ARM-7TDMI-S Rev 4 with EmbeddedICE-RT.

RM_OPT_HARDWATCHPOINT=TRUE

Enabled for cores with EmbeddedICE-RT. This device uses ARM-7TDMI-S Rev 4 with EmbeddedICE-RT.

RM_OPT_SEMIHOSTING=FALSE

This option enables or disables support for SWI semi-hosting. Semi-hosting provides code running on an ARM target use of facilities on a host computer that is running an ARM debugger. Examples of such facilities include the keyboard input, screen output, and disk I/O.

RM_OPT_SAVE_FIQ_REGISTERS=TRUE

This option determines whether the FIQ-mode registers are saved into the registers block when RealMonitor stops.

RM_OPT_READBYTES=TRUE

RM_OPT_WRITEBYTES=TRUE

RM_OPT_READHALFWORDS=TRUE

RM_OPT_WRITEHALFWORDS=TRUE

RM_OPT_READWORDS=TRUE

RM_OPT_WRITEWORDS=TRUE

Enables/Disables support for 8/16/32 bit read/write.

RM_OPT_EXECUTECODE=FALSE

Enables/Disables support for executing code from "execute code" buffer. The code must be downloaded first.

RM_OPT_GETPC=TRUE

This option enables or disables support for the RealMonitor GetPC packet. Useful in code profiling when real monitor is used in interrupt mode.

RM_EXECUTECODE_SIZE=NA

"execute code" buffer size. Also refer to RM_OPT_EXECUTECODE option.

RM_OPT_GATHER_STATISTICS=FALSE

This option enables or disables the code for gathering statistics about the internal operation of RealMonitor.

RM_DEBUG=FALSE

This option enables or disables additional debugging and error-checking code in RealMonitor.

RM_OPT_BUILDIDENTIFIER=FALSE

This option determines whether a build identifier is built into the capabilities table of RMTarget. Capabilities table is stored in ROM.

RM_OPT_SDM_INFO=FALSE

SDM gives additional information about application board and processor to debug tools.

RM_OPT_MEMORYMAP=FALSE

This option determines whether a memory map of the board is built into the target and made available through the capabilities table

RM_OPT_USE_INTERRUPTS=TRUE

This option specifies whether RMTarget is built for interrupt-driven mode or polled mode.

RM_FIFOSIZE=NA

This option specifies the size, in words, of the data logging FIFO buffer.

CHAIN_VECTORS=FALSE

This option allows RMTarget to support vector chaining through µHAL (ARM HW abstraction API).

ARM-based Microcontroller

LPC2106/2105/2104

Purchase of Philips I²C components conveys a license under the Philips' I²C patent to use the components in the I²C system provided the system conforms to the I²C specifications defined by Philips. This specification can be ordered using the code 9398 393 40011.

Definitions

Short-form specification — The data in a short-form specification is extracted from a full data sheet with the same type number and title. For detailed information see the relevant data sheet or data handbook.

Limiting values definition — Limiting values given are in accordance with the Absolute Maximum Rating System (IEC 60134). Stress above one or more of the limiting values may cause permanent damage to the device. These are stress ratings only and operation of the device at these or at any other conditions above those given in the Characteristics sections of the specification is not implied. Exposure to limiting values for extended periods may affect device reliability.

Application information — Applications that are described herein for any of these products are for illustrative purposes only. Philips Semiconductors make no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Disclaimers

Life support — These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips Semiconductors customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips Semiconductors for any damages resulting from such application.

Right to make changes — Philips Semiconductors reserves the right to make changes in the products—including circuits, standard cells, and/or software—described or contained herein in order to improve design and/or performance. When the product is in full production (status 'Production'), relevant changes will be communicated via a Customer Product/Process Change Notification (CPCN). Philips Semiconductors assumes no responsibility or liability for the use of any of these products, conveys no license or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified.

Contact information

For additional information please visit
http://www.semiconductors.philips.com. Fax: **+31 40 27 24825**

For sales offices addresses send e-mail to:
sales.addresses@www.semiconductors.philips.com.

Date of release: 10-03

Document order number: 9397 750 12239

Let's make things better.

Philips Semiconductors

PHILIPS