特点

- 集成了 ARM7TDMI® ARM® Thumb® 处理器
  - 高性能的 32 位 RISC 架构
  - 高密度的 16 位指令集
  - 性能 / 功耗 (MIPS/Watt) 的领先者
  - 嵌入式 ICE 电路仿真，支持调试通讯
- 64K 字节的片内高速 Flash 存储器，共 512 页，每页 128 字节
  - 在最坏的条件下可以 30 MHz 的速度进行单时钟周期访问，
    预取 (Prefetch) 缓冲器可以实现 Thumb 指令的优化，使处理器以最快的速度执行指令
  - 页编程时间为 4 ms，包括页自动擦除，全片擦除时间为 10 ms
  - 10,000 次的写寿命，10 年数据保持能力，扇区锁定功能， Flash 安全锁定位
  - 适合量产的快速 Flash 编程接口
- 16K 字节的片内高速 SRAM，可以在最高时钟速度下进行单时钟周期访问操作
- 存储器控制器 (MC)
  - 嵌入式 Flash 控制器，异常中断 (Abort) 状态及未对齐 (Misalignment) 检测
- 复位控制器 (RSTC)
  - 上电复位和经过工厂标定的掉电检测
  - 提供复位源信息以及给外部电路使用的复位信号
- 时钟发生器 (CKGR)
  - 低耗 RC 振荡器， 3 到 20MHz 的片上振荡器和一个 PLL
- 电源管理控制器 (PMC)
  - 可以通过软件进行电源优化，包括慢速时钟模式 (低至 500 Hz) 和空闲 (Idle) 模式
  - 三个可编程的外部时钟信号
- 先进的中断控制器 (AIC)
  - 可以单独屏蔽的、具有 8 个优先级的向量式中断源
  - 两个外部中断源和一个快速中断源，可以防止虚假 (spurious) 中断
- 调试单元 (DBGU)
  - 2 线 UART，支持调试通讯通道中断；可通过程序来禁止通过 ICE 进行访问
- 周期性间隔定时器 (PIT)
  - 20 位可编程的计数器，加上 12 位的间隔计数器
- 时间窗看门狗 (WDT)
  - 12 位受预设值 (key) 保护的可编程计数器
  - 为系统提供复位或中断信号
  - 当处理器处于调试状态或空闲模式时可以停止计数器
- 实时定时器 (RTT)
  - 32 位自由运行的具有报警功能的计数器
  - 时钟来源于片内 RC 振荡器
- 一个并行输入 / 输出控制器 (PIOA)
  - 32 个可编程的复用 I/O，每个 I/O 最多可以支持两个外设功能
  - 输入电平改变时，每个 I/O 都可以产生中断
  - 可以独立编程为开漏输出、使能上拉电阻以及同步输出
- 11 个外设数据控制器 (PDC) 通道
- 一个 USB 2.0 全速 (12 Mbps) 设备端口
  - 片上收发器，328 字节可编程的 FIFO
- 一个同步串行控制器 (SSC)
  - 每个接收器和发送器都具有独立的时钟和帧同步信号
  - 支持 I²S，支持时分多址
  - 支持 32 位数据传输的高速连续数据流功能
- 两个通用的同步 / 异步收发器 (USART)
  - 独立的波特率发生器， IrDA 红外调制 / 解调
  - 支持 ISO7816 T0/T1 智能卡，硬件握手信号，支持 RS485
  - USART1 支持全功能的调制解调器信号
- 主 / 从串行外设接口 (SPI)
  - 8 到 16 位可编程的数据长度，4 个片选线
- 一个 3 通道的 16 位定时器 / 计数器 (TC)
  - 3 个外部时钟输入端，每个通道有两个多功能 I/O 引脚
  - 倍速 PWM 发生功能，捕捉 / 波形模式，递增 / 递减计数

AT91 ARM® Thumb® 微处理器

AT91SAM7S64

初稿

本文是英文数据手册的中文翻译，其目的是方便中国用户的阅读。它无法自动跟随原稿的更新，同时也可能存在翻译上的错误。读者应该以英文原稿为参考以获得更准确的信息。

6070A–ATARM–07-Jun-05

- 一个 4 通道的 16 位 PWM 控制器 (PWMC)
- 一个两线接口 (TWI)
  - 只支持主机模式，支持所有的 Atmel 两线 EEPROM
- 一个 8 通道的 10 位模数转换器，其中 4 个通道与数字 I/O 复用
- IEEE 1149.1 JTAG 边界扫描支持所有的数字引脚
- 5V 兼容的 I/O，包括 4 个高达 16 mA 的大电流驱动 I/O
- 电源
  - 片上 1.8V 电压调节器，可以为内核及外部元件提供高达 100 mA 的电流
  - 为 I/O 口线提供电源的 3.3V VDDIO，以及独立的为 Flash 供电的 3.3V VDDFLASH
  - 内核电源为 1.8V VDDCORE，并具有掉电检测 (BoD) 功能
- 全静态操作：极限条件下 (1.65V， 85° C) 高达 55 MHz
- 封装为 64 脚的 LQFP

描述

AT91SAM7S64 是 Atmel 32 位 ARM RISC 处理器小引脚数 Flash 微处理器家族的一员。它拥有 64K 字节的高速 Flash 和 16K 字节的 SRAM，丰富的外设资源，包括一个 USB 2.0 设备，使外部器件数目减至最低的完整系统功能集。这个芯片是那些正在寻求额外处理能力和更大存储器的 8 位处理器用户的理想选择。

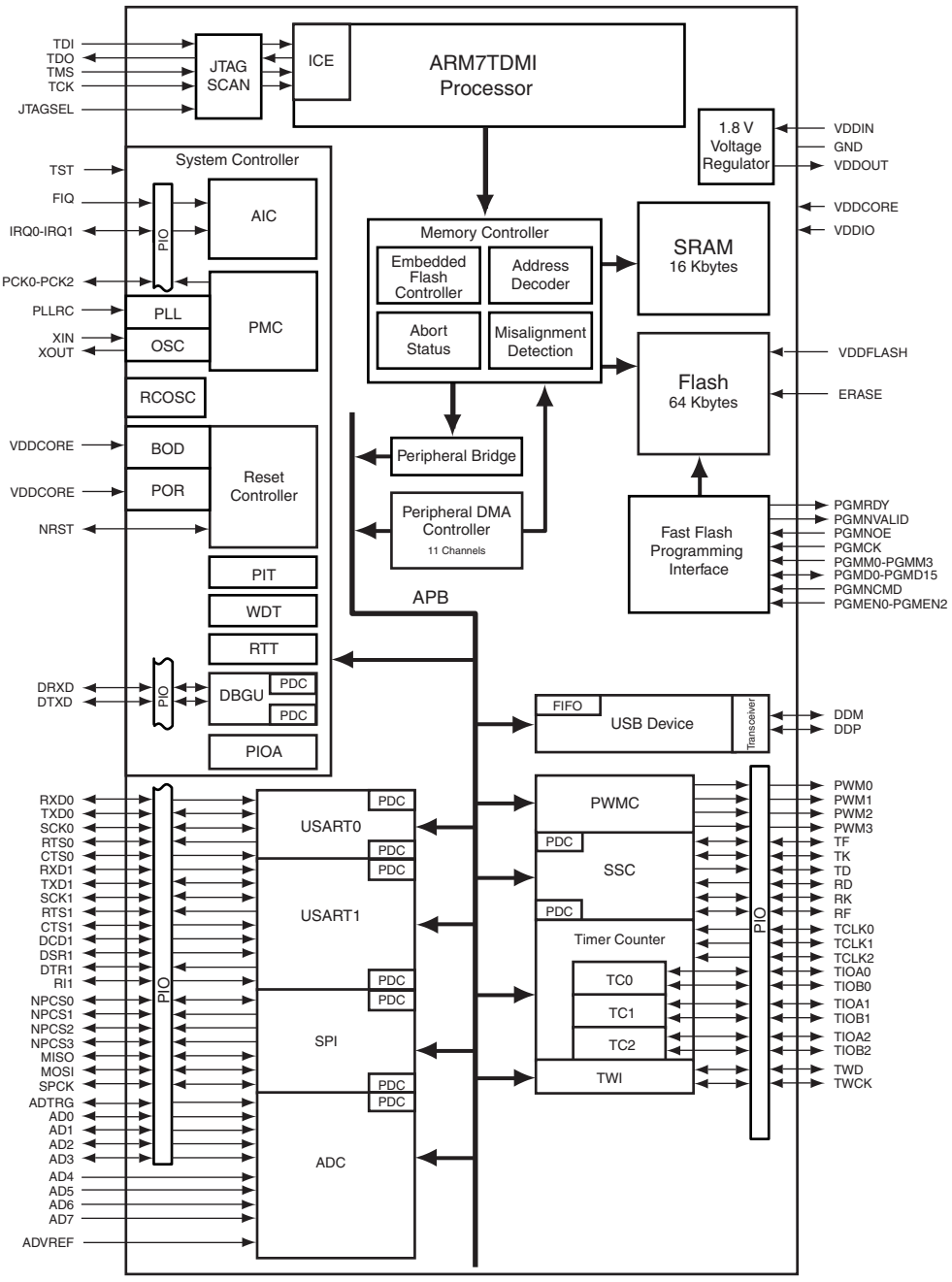
Flash 存储器可以通过 JTAG-ICE 进行编程，或者是在贴装之前利用编程器的并行接口进行编程。锁定位可以防止固件不小心被改写，而安全锁定位则可以保护固件的安全。

AT91SAM7S64 的复位控制器可以管理芯片的上电顺序以及整个系统。BOD 和看门狗则可以监控器件是否正确工作。

AT91SAM7S64 是一个通用处理器。它集成了 USB 设备端口，使得它成为连接 PC 或手机的外设应用的理想芯片。极具竞争力的性价比进一步拓展了它在低成本、大产量的消费类产品中的应用。

方框图

Figure 1. AT91SAM7S64 框图



信号说明

Table 1. 信号说明列表

信号名称	功能	类型	有效电平	说明
电源				
VDDIN	电压调节器电源输入端	电源		3.0V - 3.6V
VDDOUT	电压调节器输出	电源		1.85V，标称值
VDDFLASH	Flash 存储器的电源	电源		3.0V - 3.6V
VDDIO	I/O 电源	电源		3.0V - 3.6V
VDDCORE	内核电源	电源		1.65V - 1.95V
VDDPLL	PLL	电源		1.65V - 1.95V
GND	地	地		
时钟，振荡器和 PLL				
XIN	主时钟振荡器输入	输入		
XOUT	主时钟振荡器输出	输出		
PLLRC	PLL 滤波器	输入		
PCK0 - PCK2	可编程的时钟输出	输出		
ICE 和 JTAG				
TCK	测试时钟	输入		没有上拉电阻
TDI	测试数据输入	输入		没有上拉电阻
TDO	测试数据输出	输出		
TMS	测试模式选择	输入		没有上拉电阻
JTAGSEL	JTAG 选择	输入		下拉电阻
Flash 存储器				
ERASE	Flash 和 NVM 配置位擦除命令	输入	高电平	下拉电阻
复位 / 测试				
NRST	处理器复位	I/O	低电平	上拉电阻
TST	测试模式选择	输入		下拉电阻
调试单元				
DRXD	调试数据接收	输入		
DTXD	调试数据发送	输出		
AIC				
IRQ0 - IRQ1	外部中断输入	输入		
FIQ	快速中断输入	输入		

Table 1. 信号说明列表

信号名称	功能	类型	有效电平	说明
PIO				
PA0 - PA31	并行 IO 控制器 A	I/O		复位时为带上拉电阻的输入端
USB Device Port				
DDM	USB 设备端口数据 -	模拟		
DDP	USB 设备端口数据 +	模拟		
USART				
SCK0 - SCK1	串行时钟	I/O		
TXD0 - TXD1	发送数据	I/O		
RXD0 - RXD1	接收数据	输入		
RTS0 - RTS1	请求发送	输出		
CTS0 - CTS1	清零后发送	输入		
DCD1	数据载波检测	输入		
DTR1	数据终端准备好	输出		
DSR1	数据设备准备好	输入		
RI1	振铃指示	输入		
同步串行控制器				
TD	发送数据	输出		
RD	接收数据	输入		
TK	发送时钟	I/O		
RK	接收时钟	I/O		
TF	发送帧同步	I/O		
RF	接收帧同步	I/O		
定时器 / 计数器				
TCLK0 - TCLK2	外部时钟输入	输入		
TIOA0 - TIOA2	I/O 口线 A	I/O		
TIOB0 - TIOB2	I/O 口线 B	I/O		
PWM 控制器				
PWM0 - PWM3	PWM 通道	输出		
SPI				
MISO	主机输入，从机输出	I/O		
MOSI	主机输出，从机输入	I/O		
SPCK	SPI 串行时钟	I/O		
NPCS0	SPI 外设片选 0	I/O	低电平	
NPCS1-NPCS3	SPI 外设片选 1 到 3	输出	低电平	

Table 1. 信号说明列表

信号名称	功能	类型	有效电平	说明
两线接口				
TWD	串行数据	I/O		
TWCK	串行时钟	I/O		
模数转换器				
AD0-AD3	模拟输入	模拟		复位时为带上拉电阻的数字输入端口
AD4-AD7	模拟输入	模拟		模拟输入
ADTRG	ADC 触发	输入		
ADVREF	ADC 参考电压	模拟		
快速 Flash 编程接口				
PGMEN0-PGMEN1	编程使能	输入		
PGMM0-PGMM3	编程模式	输入		
PGMD0-PGMD15	编程的数据	I/O		
PGMRDY	编程结束	输出	高电平	
PGMNVALID	数据方向	输出	低电平	
PGMNOE	编程过程中读数据	输入	低电平	
PGMCK	编程时钟	输入		
PGMNCMD	编程命令	输入	低电平	

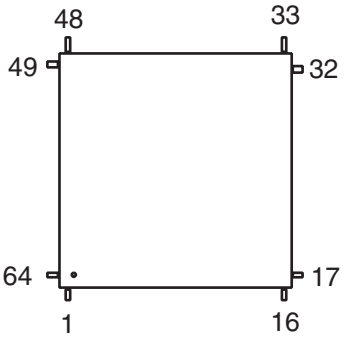
封装和引脚排列

AT91SAM7S64 的封装为 64 脚的 LQFP。

64脚LQFP封装的机械形状

Figure 2 给出了 64 脚 LQFP 封装的定位。具体的机械尺寸在机械特性一节有详细说明。

Figure 2. 64 脚 LQFP 封装引脚 (顶视图)



引脚排列

Table 2. AT91SAM7S64 引脚排列 (64 脚 LQFP 封装)

1	ADVREF	17	GND	33	TDI	49	TDO
2	GND	18	VDDIO	34	PA6/PGMNOE	50	JTAGSEL
3	AD4	19	PA16/PGMD4	35	PA5/PGMRDY	51	TMS
4	AD5	20	PA15/PGM3	36	PA4/PGMNCMD	52	PA31
5	AD6	21	PA14/PGMD2	37	PA27/PGMD15	53	TCK
6	AD7	22	PA13/PGMD1	38	PA28	54	VDDCORE
7	VDDIN	23	PA24/PGMD12	39	NRST	55	ERASE
8	VDDOUT	24	VDDCORE	40	TST	56	DDM
9	PA17/PGMD5/AD0	25	PA25/PGMD13	41	PA29	57	DDP
10	PA18/PGMD6/AD1	26	PA26/PGMD14	42	PA30	58	VDDIO
11	PA21/PGMD9	27	PA12/PGMD0	43	PA3	59	VDDFLASH
12	VDDCORE	28	PA11/PGMM3	44	PA2	60	GND
13	PA19/PGMD7/AD2	29	PA10/PGMM2	45	VDDIO	61	XOUT
14	PA22/PGMD10	30	PA9/PGMM1	46	GND	62	XIN/PGMCK
15	PA23/PGMD11	31	PA8/PGMM0	47	PA1/PGMEN1	63	PLLRC
16	PA20/PGMD8/AD3	32	PA7/PGMNVALID	48	PA0/PGMEN0	64	VDDPLL

电源的考虑

电源

AT91SAM7S64 有 6 种类型的电源输入引脚以及一个集成的电源调节器，使得器件可以工作于单一电压。这 6 种电源引脚类型为：

- VDDIN：电压调节器的电源输入。输入电压范围是 3.0V 到 3.6V，标称值为 3.3V。如果不用电压调节器，则 VDDIN 应该连接到 GND。
- VDDOUT：电压调节器的输出， 1.8V。
- VDDIO：I/O 及 USB 的电源。支持电压范围为 3.0V 到 3.6V，标称值为 3.3V。
- VDDFLASH：为 Flash 部分地提供电源，而且是 Flash 正确工作的先决条件。电压范围为 3.0V 到 3.6V，标称值为 3.3V。
- VDDCORE：芯片逻辑部分的电源。电压范围从 1.65V 到 1.95V，典型值为 1.8V。可以通过解耦电容连接到 VDDOUT 引脚。 VDDCORE 是器件内核，包括 Flash 正确工作的前提。
- VDDPLL：振荡器和 PLL 的电源。可以直接连接到 VDDOUT。

各个输入电源并没有独立的地回路引脚。因此 GND 与系统地平面的连接应尽可能短。

功耗

在 25°C 时，VDDCORE 的静态电流小于 60 µA，包括 RC 振荡器、电压调节器和上电复位。使能掉电复位 BOD 将额外增加 20 µA 的静态电流。

全速工作且运行不基于 Flash 时 VDDCORE 的动态功耗 小于 50 mA。若程序在 Flash 上运行则 VDDFLASH 的电流不超过 10 mA。

电压调节器

AT91SAM7S64 有一个由系统控制器管理的电压调节器。

在正常模式下，电压调节器消耗的静态电流还不到 100 µA，而输出电流则高达 100 mA。

电压调节器支持低功耗模式。在此模式下它只消耗不到 20 µA 的静态电流，输出电流可达 1 mA。

VDDOUT 必须有足够的解耦电容以减少纹波和防止振荡。最好的方法是并联两个电容于 VDDOUT 和 GND 之间：一个 470 pF (或 1 nF) NPO 材质的电容，尽量靠近芯片；另一个是 2.2 µF (或 3.3 µF) X7R 材质的电容。

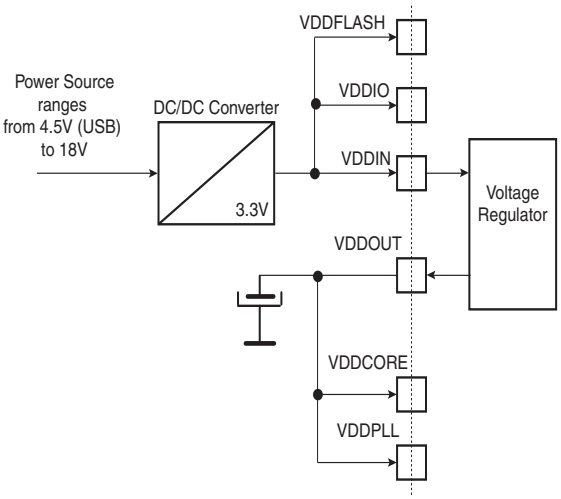
VDDIN 也需要足够的解耦来提高启动稳定性以及减少电压降。输入电容也需要尽量靠近芯片。例如可以将两个电容并联在一起：100 nF 的 NPO 电容和 4.7 µF 的 X7R 电容。

典型的电源连接图

3.3V 单电源

AT91SAM7S64 支持 3.3V 单电源模式。片内电压调节器连接到 3.3V，输出则连接到 VDDCORE 和 VDDPLL。 Figure 3 给出了通过 USB 总线供电的线路图。

Figure 3. 3.3V 单电源供电系统



I/O 的考虑

JTAG 引脚

TMS、TDI 和 TCK 都是施密特触发器型的输入引脚。TMS 和 TCK 与 5V 兼容，TDI 则不是。TMS、 TDI 和 TCK 都没有上拉电阻。

TDO 为输出引脚，输出电平可达 VDDIO，没有上拉电阻。

引脚 JTAGSEL 拉高时选择 JTAG 边界扫描功能。此引脚集成了阻值约为 15 k Ω 的片内下拉电阻，所以在正常工作模式下可以悬空。

测试引脚

TST 用于生产测试，或是拉高以进入快速编程模式。TST 集成了阻值约为 15 kΩ 的片内下拉电阻，所以在正常工作模式下可以悬空。

为了使能快速编程模式，引脚 TST 和 PA0、 PA1 都需要拉高。

TST 为高时，若 PA0 或 PA1 为 0 将导致不可预测的结果。

复位引脚

NRST 是双向引脚。它可以被片内的复位控制器拉低，从而为外部器件产生复位信号；也可以被外部电路拉低以复位处理器。复位脉冲没有持续时间的限制，复位控制器可以保证产生最小长度的脉冲。从而可以在引脚 NRST 上简单地连接一个揿键开关作为用户的系统复位控制，同时也可以利用 NRST 信号来复位系统的其他原器件。

NRST 引脚有一个上拉电阻连接到 VDDIO。

ERASE 引脚

引脚 ERASE 用于 Flash 及其一些 NVM 位的擦除。引脚具有阻值约为 15 k Ω 的下拉电阻，所以在正常工作模式下可以悬空。

PIO 控制器 A 端口

I/O 口线 PA0-PA31 与 5V 兼容，且每个 I/O 都具有可编程的上拉电阻。通过 PIO 控制器可以对每一个 I/O 的上拉电阻进行单独控制。

与 5V 兼容意味着 I/O 可以输出 VDDIO 的电平，而输入则可以高达 5.5V。但此时不能使能 I/O 的上拉电阻，否则将导致不可预测的结果。特别是在复位时一定要注意，因为在复位阶段所有的 I/O 都缺省为带上拉电阻的输入引脚。

I/O 驱动电平

PA0-PA3 可以输出大电流。每个 I/O 都可以驱动高达 16 mA 的电流。

其余的 I/O 只能输出 8 mA 的电流。

要注意的是，所有 I/O 输出的电流之和不能超过 150 mA。

处理器和结构

ARM7TDMI 处理器

- 基于 ARMv4T 冯 - 诺依曼结构的 RISC 处理器
  - 运行速度可达 55 MHz， 0.9 MIPS/MHz 的性能
- 两个指令集
  - ARM® 高性能 32 位指令集
  - Thumb® 高代码密度 16 位指令集
- 3 级流水线结构
  - 指令获取 (F)
  - 指令解码 (D)
  - 执行 (E)

调试和测试特点

- 集成的片上仿真器
  - 两个观察点 (watchpoint) 单元
  - 通过 JTAG 协议访问测试访问端口 TAP
  - 调试通讯通道
- 调试单元
  - 两线 UART
  - 可以处理调试通讯通道中断
  - 芯片 ID 寄存器
- IEEE1149.1 JTAG 边界扫描支持所有的数字引脚

存储器控制器

- 总线仲裁
  - 处理来自 ARM7TDMI 和外设数据控制器的请求
- 地址译码器可以提供如下片选信号
  - 3 个 1M 字节的片内存储区
  - 一个 256M 字节的片内外设区
- 仲裁状态寄存器
  - 保存了引发仲裁的来源、类型以及其他所有参数
  - 通过检测被破坏的指针以方便调试
- 对齐 (alignment) 检测
  - 访问数据时的对齐检查
  - 发生未对齐情况时产生异常中断
- 重映射 (Remap) 命令
  - 将 SRAM 映射到片内非易失性存储器 (NVM) 的位置
  - 允许例外向量的动态处理
- 嵌入式 Flash 控制器
  - 嵌入式 Flash 接口，最多可有 3 个可编程的等待周期
  - 预取缓冲器，用于缓冲及预留 16 位请求，从而减少等待周期
  - 受预设值保护的编程、擦除和锁定 / 解锁定序器
  - 存储器擦除、编程和锁定操作都只需要一个命令
  - 执行被禁止的操作将引发中断

外设数据控制器

- 处理外设与存储器之间的数据传输
- 11 个通道
  - 每个 USART 有两个
  - 调试单元有一个
  - 串行同步控制器 (SSC) 有两个
  - SPI 有两个
  - 模数转换器有一个
- 低的总线仲裁开销
  - 从存储器到外设的传输只需要一个主时钟周期
  - 从外设到存储器的传输只需要两个主时钟周期
- “ 下一个指针”管理减少了中断时间

存储器

- 64K 字节 Flash
  - 512 页，每页 128 字节
  - 快速的访问时间，在最坏的条件下访问周期可达到 30 MHz
  - 页编程时间：4 ms，包括页自动擦除时间
  - 没有自动擦除操作的页编程时间：2 ms
  - 全片擦除时间：10 ms
  - 10,000 次写寿命， 10 年数据保存时间
  - 16 个锁定位，每个保护一个扇区 (每个扇区包含 32 页)
  - 保护 Flash 内容安全的保护模式
- 16K 字节的快速 SRAM
  - 在全速工作时仍然可以单时钟进行访问

存储器映射

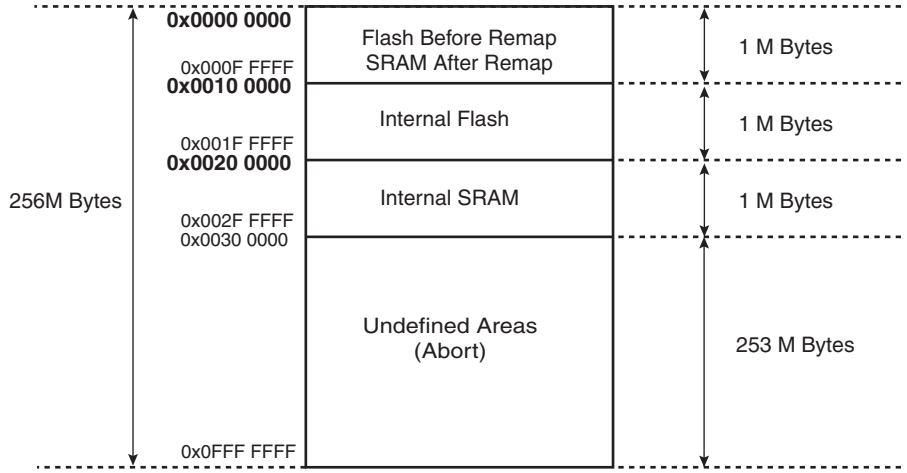
片内 SRAM

AT91SAM7S64 有 16K 字节的高速 SRAM。芯片复位后，直到执行 Remap 命令，SRAM 的访问地址为 0x0020 0000。重映射之后， SRAM 的访问地址变为 0x0。

片内 Flash

AT91SAM7S64 有 64K 字节的 Flash。在任何时候它的访问地址都是 0x0010 0000。此外，在芯片复位之后， Remap 之前，也可以从 0x0 进行访问。

Figure 4. 片内存储器映射



片内 Flash

Flash 概览

AT91SAM7S64的Flash组织为512页，每页128字节。全部65,536字节以32位字的方式组织在一起。

Flash 包含了一个 128 字节的写缓冲区，可以 32 位的接口进行访问。

片内复位单元和低电压检测器 BOD可以帮助 Flash 在电压变化时防止代码毁坏，即使在最坏的情况之下。

片内 Flash 控制器

片内 (嵌入式)Flash 控制器 (EFC) 管理系统各个主机执行的存储器访问。它控制对 Flash 的读访问以及对写缓冲区的写访问。它还包括了一个用户接口，映射到 APB 的存储器控制器。用户接口允许：

- 编程 Flash 的访问参数 (等待周期的个数，时序，等等)
- 启动诸如全片擦除、页擦除、页写、 NVM 位设置 / 清除等命令
- 获取上一个命令的结束状态
- 获取错误状态
- 上一个命令结束或发生错误时产生中断

片内 Flash 控制器还提供双 32 位预取缓冲器以优化对 Flash 存储器的 16 位访问。这对运行于 Thumb 模式的处理器特别有效。

锁定区域

EFC管理着16个锁定位以保护flash的16个区，防止这些区域被意外地擦除或编程。每个锁定区域包含 32 页，共 4K 字节。

如果对已经被锁住的区域进行擦除或编程，这些命令将终止，同时 EFC 激发一个中断。

通过 EFC 用户接口可以对 16 个 NVM 位实行软件编程。“ 设置锁定位” 命令启动保护操作；“ 清除锁定位” 命令解除锁定区域的锁定状态。

将 ERASE 拉高将清除所有的锁定位，从而将全部的 Flash 解锁。

安全位的特点

AT91SAM7S64 有一个安全位。它是一个特殊的 NVM 位。当安全位使能时，对 Flash 的所有访问，包括通过 ICE 接口或快速 Flash 编程接口，都被禁止。从而 保护了 Flash 的内容。

这个安全位只能通过 EFC 用户接口的“ 设置安全位” 命令来使能。而禁止安全位只能通过将 ERASE 引脚拉高，将整个 flash 全部擦除。在安全位为禁止状态的情况下，对 flash 的所有操作都可以进行。

很重要的一点是，拉高 ERASE 引脚的时间必须大于 50 ms。

由于 ERASE 集成了下拉电阻，在正常工作模式下这个引脚可以悬空。不过将它直接连接到 GND 也是安全的。

非易失性掉电检测控制

掉电检测 (BOD) 由两个通用的 NVM (GPNVM) 位控制。因此即使没有了电源，掉电检测仍然可以保持用户的定义。

两个 GPNVM 位的清除和设置分别通过 EFC 用户接口的“ 清除通用 NVM 位” 命令和“ 设置通用 NVM 位” 命令来实现。

- GPNVM 位 0 用于控制掉电检测的使能。设置 GPNVM0 将使能 BOD，清除它即禁止 BOD。拉高 ERASE 将清除 GPNVM0，从而禁止 BOD。
- GPNVM 位 1 控制掉电检测信号是否可以用于系统复位。置位 GPNVM1 使能这个功能，清零 GPNVM1 则禁止掉电检测信号复位整个芯片。拉高 ERASE 将禁止掉电检测复位。

标定位

8个NVM位用于标定掉电检测器及电压调节器。这些位的配置在出厂之前完成，用户不能进行修改。 ERASE 引脚的状态对标定位没有影响。

快速 Flash 编程接口 FFPI

快速 Flash 编程接口允许用户通过串行 JTAG 接口或实现了握手信号的并行口对 Flash 进行快速编程。这样就可以通过符合市场标准的工业编程器进行批量编程。

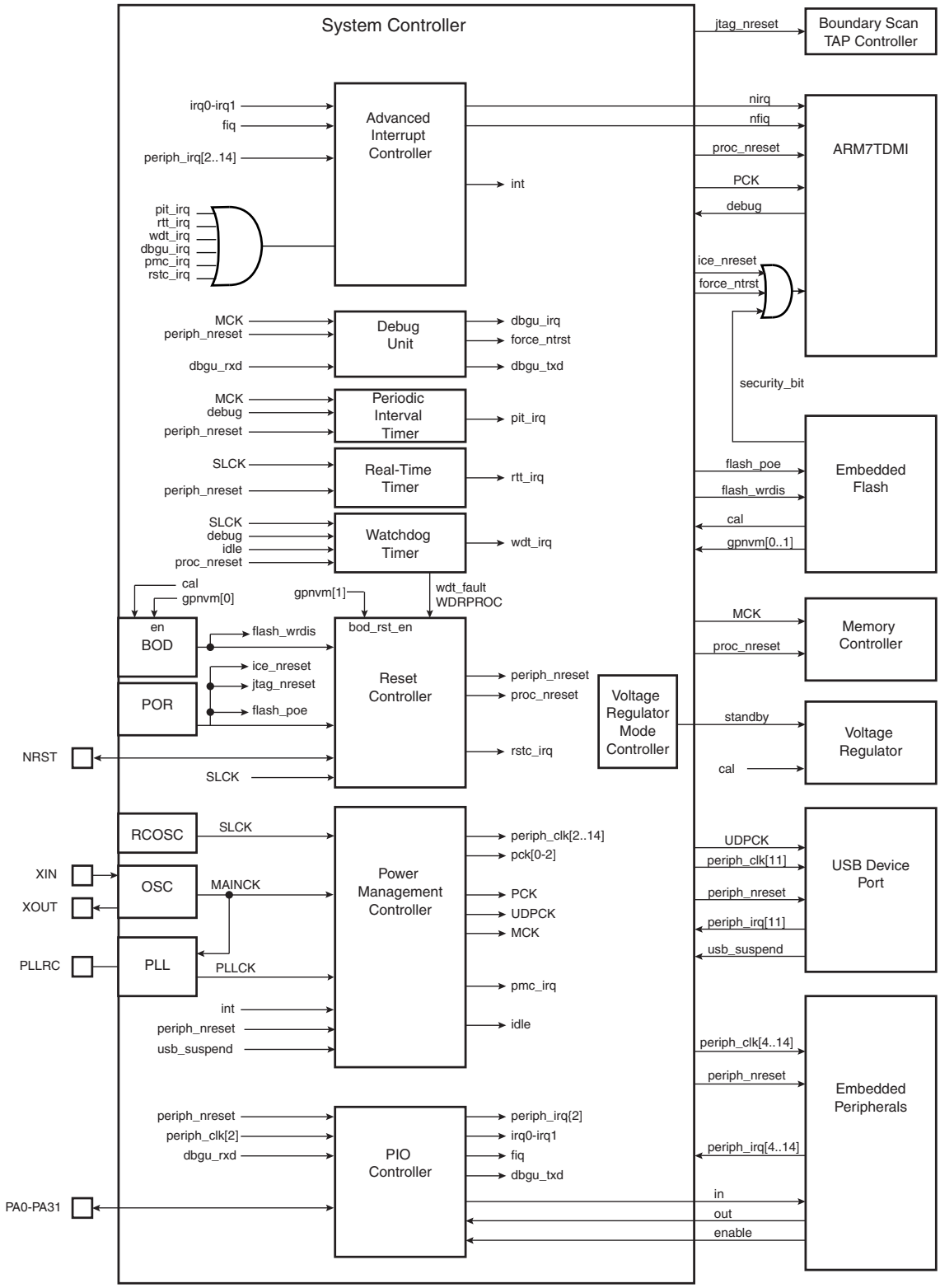
FFPI 支持读、页编程、页擦除、全片擦除、锁定、解锁和保护命令。

拉高引脚 TST、 PA0 和 PA1 即使能 FFPI，并使芯片进入快速编程模式。

系统控制器

系统控制器管理处理器所有重要的模块：中断、时钟、电源、时序、调试以及复位。

Figure 5. 系统控制器框图

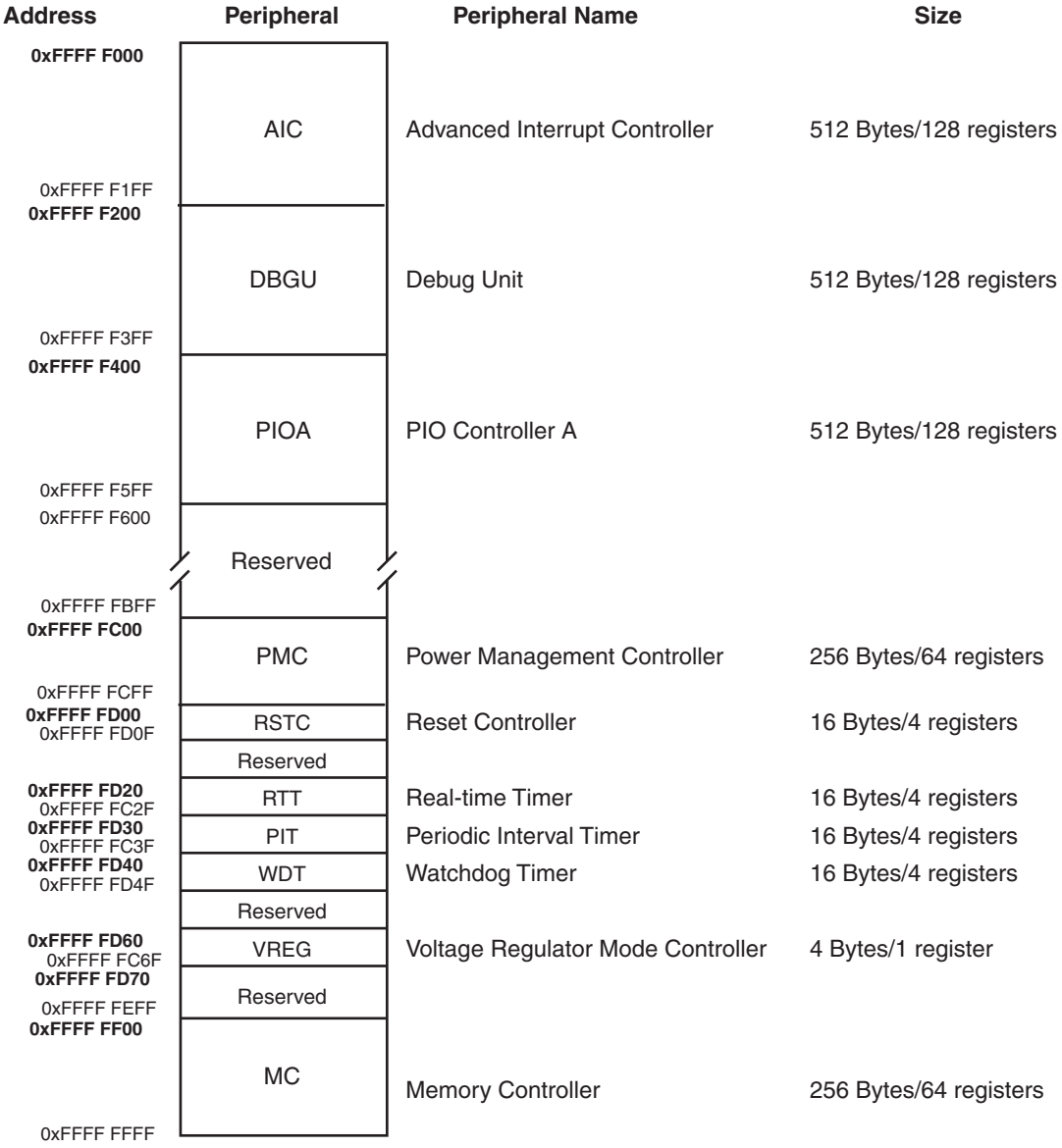


系统控制器映射

系统控制器外设映射到 4K 字节的最高地址空间，界于 0xFFFF F000 和 0xFFFF FFFF。

Figure 6 给出了系统控制器的映射情况。要注意的是，存储器控制器配置用户接口也映射于这一地址区间。

Figure 6. 系统控制器映射



复位控制器

复位控制器包括上电复位和掉电检测复位。它可以给出上一次复位的状态，指明上一次复位是上电复位、软件复位、用户复位、看门狗复位还是掉电检测复位。此外，它还控制内部复位以及 NRST 引脚的输出，并控制 NRST 脉冲的长度以满足系统要求。

掉电检测及上电复位

AT91SAM7S64 嵌入了掉电检测电路和上电复位单元。两个电路都由 VDDCORE 供电，并反过来监视 VDDCORE。两个信号都输送到 Flash，以防止在上电和下电的过程中，或是电压跳变的过程中， Flash 中的代码被损毁。

上电复位单元具有有限精度的门限值，大约是 1.5V。在 VDDCORE 达到这个电平之前其输出一直保持为低。这个信号输送到复位控制器，实现对芯片的重新初始化。

掉电检测器监视 VDDCORE 的电平，将其与固定的触发电平进行比较。它可以在大多数情况下保证系统安全工作，并保证在 VDDCORE 发生跳变时存储器里的代码不被损毁。

掉电检测器只监视 VDDCORE。这是因为 VDDFLASH 或其他电源的跌落对 Flash 的意外改写没有影响。

掉电检测器使能后，一旦 VDDCORE 下降到触发电平 (Vbot-，定义为 Vbot - hyst/2)，掉电检测输出立即被激活。

当 VDDCORE 上升到触发电平 (Vbot+，定义为 Vbot + hyst/2)，复位信号即被释放。掉电检测器只监视 VDDCORE 跌落到门限值以下，持续时间大于 1µs 的情况。

为了防止电压尖峰的影响，门限电压有大约为 50 mV 的滞后值。门限电压的典型值为 1.68V，精度为 ± 2%。

掉电检测器是低功耗单元，只消耗大约 20 µA 的静态电流。为了进一步降低系统静态功耗，还可以将其关闭。此时它消耗的静态电流小于 1µA。关闭掉电检测器是通过配置 GPNVM0 来实现的。

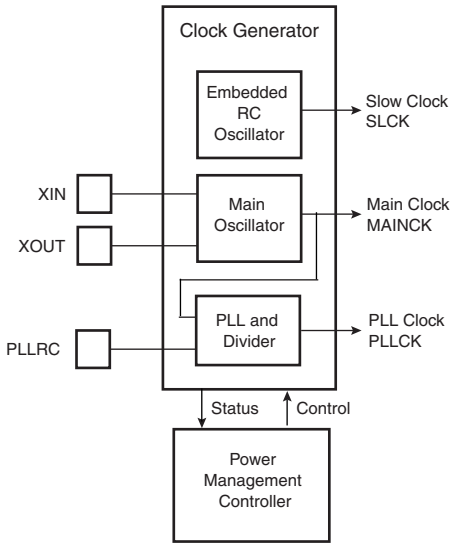
时钟发生器

时钟发生器包括一个低功耗 RC 振荡器，一个主振荡器和一个 PLL。其特性为：

- RC 振荡器的频率范围是 22 KHz 到 42 KHz
- 主振荡器的频率范围是 3 到 20 MHz
- 主振荡器可以被旁路
- PLL 输出范围是 80 到 200 MHz

它提供了 SLCK、 MAINCK 以及 PLLCK。

Figure 7. 时钟发生器框图



电源管理控制器

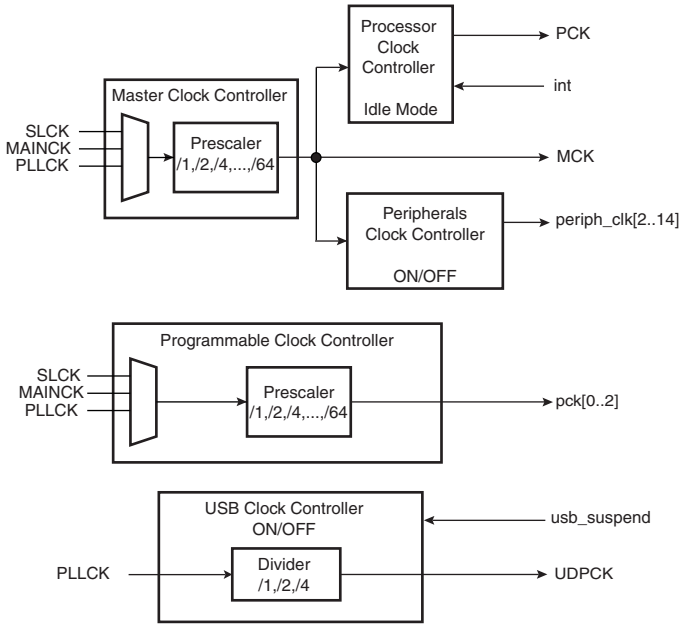
电源管理控制器利用时钟发生器的输出提供 :

- 处理器时钟 PCK
- 主时钟 MCK
- USB 时钟 UDPCK
- 所有独立可控的外设时钟
- 3 个可编程的时钟输出

通过编程，主时钟 (MCK) 可以从几百赫兹一直到芯片的最大工作频率。

处理器进入空闲模式等待中断发生时，处理器时钟 (PCK) 被关闭，从而降低器件功耗。

Figure 8. 电源管理控制器框图



先进的中断控制器

- 控制 ARM 处理器的中断线 (nIRQ 和 nFIQ)
- 可以单独屏蔽的向量化中断源
  - 中断源 0 为快速中断输入 (FIQ)
  - 中断源 1 为系统外设 (RTT， PIT， EFC， PMC， DBGU，等)
  - 其他中断源控制外设中断或外部中断
  - 可编程的边沿触发或电平敏感的内部中断源
  - 可编程的正 / 负边沿触发或高 / 低电平敏感的外部中断源
- 8 级优先级控制器
  - 驱动处理器的正常中断
  - 处理中断源的优先级
  - 高优先级中断可以插入正在执行的低优先级中断
- 向量化
  - 优化中断服务例程的跳转及执行
  - 每个中断源都有一个 32 位的向量寄存器
  - 中断向量寄存器读取对应的当前中断向量

- 保护模式
  - 通过禁止自动操作简化调试
- 强制更新为快速中断
  - 允许将任意一个中断源重定义为快速中断
- 通用的中断屏蔽
  - 使处理器保持与事件的同步，同时不引发中断

调试单元

- 构成元素：
  - 两线 UART
  - 支持调试通讯通道 (DCC) 的接口
  - 一组芯片 ID 寄存器
  - 一个禁止 ICE 访问的接口
- 两线 UART
  - 实现与 USART 兼容的特性
  - 可编程的波特率发生器
  - 奇偶校验、帧错误和超速错误
  - 自动回应、本地回环和远程回环通道模式
- 支持调试通讯通道
  - 输出了 ARM 处理器的 COMMRX 和 COMMTX 信号
- 芯片 ID 寄存器
  - 识别器件的版本，片内存储器的大小及外设集
  - 芯片 ID 为 0x27090540 (版本 0)

周期性间隔定时器

- 20 位可编程的计数器，外加 12 位的间隔计数器

看门狗定时器

- 12 位受预设值保护的可编程计数器，计数时钟为经过预分频的 SLCK
- 为系统提供复位或中断信号
- 处理器处于调试状态或空闲模式时可以停止计数器

实时定时器

- 32 位自由运行的计数器，带闹铃功能，计数时钟为经过预分频的 SLCK
- 可编程的 16 位预分频器用于 SLCK 精度补偿

PIO 控制器

- 一个 PIO 控制器，掌管 32 个 I/O 口线
- 通过设置 / 清除寄存器实现对 I/O 的完全控制
- 每个 I/O 都被两个外设功能所复用
- 每一个 I/O 口 (不论是分配给外设使用还是作为通用 I/O) 都支持
  - 输入电平变化中断
  - 半个时钟周期的尖峰滤波器
  - 多种驱动选择，可以工作于开漏状态
  - 可编程的上拉电阻
  - 引脚数据状态寄存器给出了任意时候引脚上的电平
- 同步输出，设置和清零几个 I/O 口线只需要一个写操作

电压调节器控制器

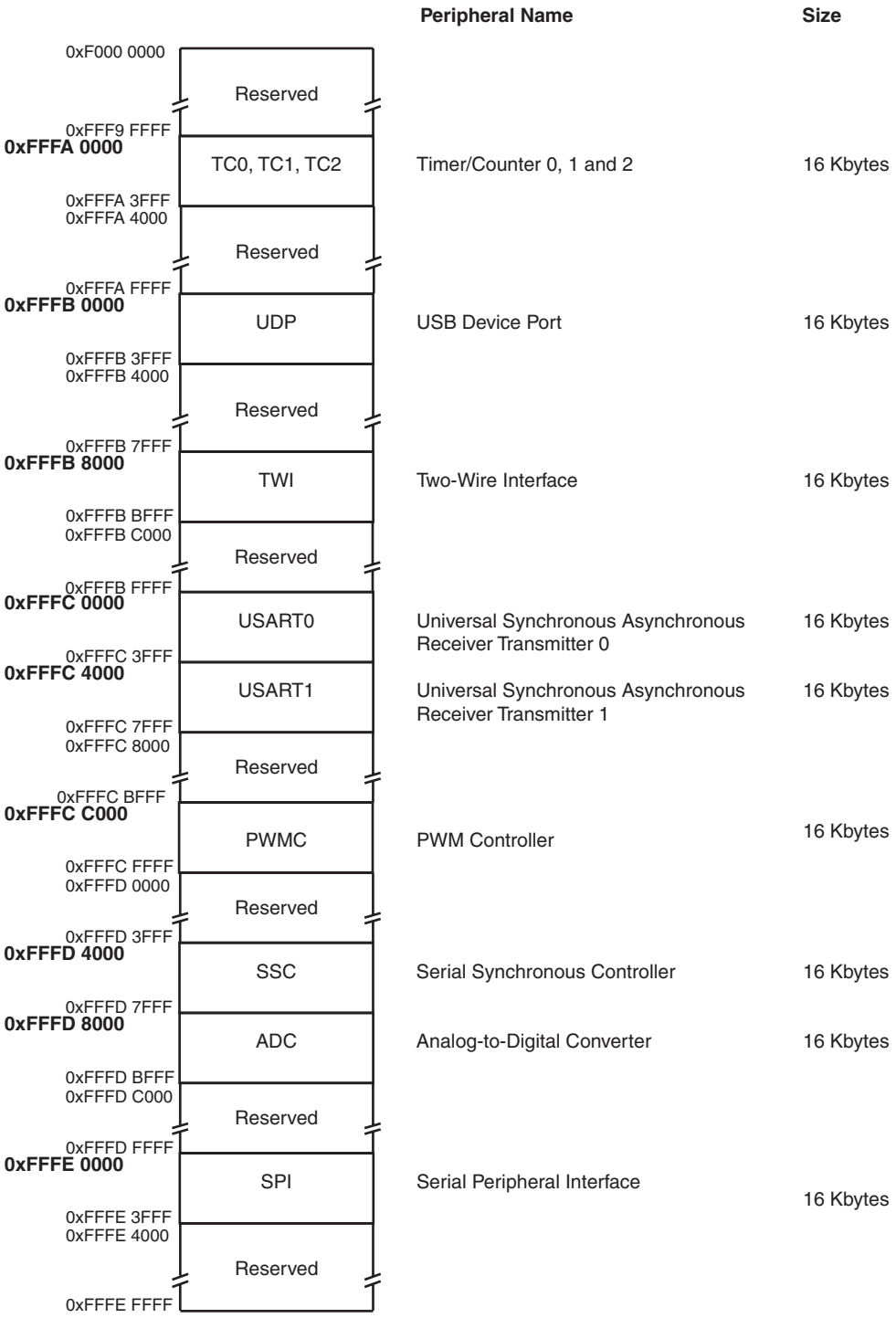
这个控制器的作用是选择电压调节器的工作模式 : 正常模式 (位 0 清零) 或 Standby 模式 (位 0 置位)。

外设

外设映射

每个外设都分配了 16K 字节的地址空间。

Figure 9. 用户外设映射



外设功能在 PIO 口线上的复用

AT91SAM7S64 的 PIO 控制器 PIOA 复用了 I/O 口线作为外设功能。

PIO控制器A控制32个口线。每个口线都可以分配为两个外设功能的一个 ,A或B。其中的一些还可以复用为 ADC 控制器的模拟输入。

Table 3 on page 25 说明了 PIO 控制器 A 的 I/O 口线是如何被外设 A ,B 或模拟输入所复用的。“ 功能 ” 及 “ 说明 ” 这两列主要是方便用户针对具体应用做注释以跟踪各个引脚是如何被分配使用的。

要注意某些只能作为输出的外设功能在表中被复制了多次。

复位时所有的并行 I/O 都配置为输入，且上拉电阻使能。

PIO 控制器 A 引脚复用

Table 3. PIO 控制器 A 引脚复用

PIO 控制器 A				应用	
I/O	外设 A	外设 B	说明	功能	说明
PA0	PWM0	TIOA0	大电流驱动		
PA1	PWM1	TIOB0	大电流驱动		
PA2	PWM2	SCK0	大电流驱动		
PA3	TWD	NPCS3	大电流驱动		
PA4	TWCK	TCLK0			
PA5	RXD0	NPCS3			
PA6	TXD0	PCK0			
PA7	RTS0	PWM3			
PA8	CTS0	ADTRG			
PA9	DRXD	NPCS1			
PA10	DTXD	NPCS2			
PA11	NPCS0	PWM0			
PA12	MISO	PWM1			
PA13	MOSI	PWM2			
PA14	SPCK	PWM3			
PA15	TF	TIOA1			
PA16	TK	TIOB1			
PA17	TD	PCK1	AD0		
PA18	RD	PCK2	AD1		
PA19	RK	FIQ	AD2		
PA20	RF	IRQ0	AD3		
PA21	RXD1	PCK1			
PA22	TXD1	NPCS3			
PA23	SCK1	PWM0			
PA24	RTS1	PWM1			
PA25	CTS1	PWM2			
PA26	DCD1	TIOA2			
PA27	DTR1	TIOB2			
PA28	DSR1	TCLK1			
PA29	RI1	TCLK2			
PA30	IRQ1	NPCS2			
PA31	NPCS1	PCK2			

外设标识符

AT91SAM7S64 拥有很多外设。Table 4 给出了各种外设的标识符。中断控制器用标识符来控制外设中断，而电源管理控制器则用标识符来控制外设时钟。

Table 4. 外设标识符

外设 ID	外设助记符	外设名称	外部中断
0	AIC	先进的中断控制器	FIQ
1	SYSIRQ(1)	系统中断	
2	PIOA	并行 I/O 控制器 A	
3	保留		
4	ADC(1)	模数转换器	
5	SPI	串行外设接口	
6	US0	USART 0	
7	US1	USART 1	
8	SSC	同步串行接口	
9	TWI	两线接口	
10	PWMC	PWM 控制器	
11	UDP	USB 设备端口	
12	TC0	定时器 / 计数器 0	
13	TC1	定时器 / 计数器 1	
14	TC2	定时器 / 计数器 2	
15 - 29	保留		
30	AIC	先进的中断控制器	IRQ0
31	AIC	先进的中断控制器	IRQ1

Note: 1. 对电源管理控制器(PMC)时钟置位/清除寄存器的SYSIRQ以及ADC进行置位操作不会引起任何作用。系统控制器总是有时钟驱动的。进行第一次转换时 ADC 时钟自动启动。在休眠模式下，每一次转换结束后 ADC 时钟自动停止。

串行外设接口 SPI

- 支持与其他串行设备的通讯
  - 4 个片选线，在解码芯片的协助下可以与多达 15 个器件进行通讯
  - 串行存储器，如 DataFlash® 和 3 线 EEPROM
  - 串行外设，如 ADC， DAC， LCD 控制器， CAN 控制器和传感器等
  - 外部协处理器
- 主 / 从 SPI 总线
  - 每个片选线都支持 8 到 16 位可编程的数据长度
  - 每个片选线都支持可编程的信号相位和极性
  - 每个片选线都支持可编程的连续数据帧间传输延迟，以及时钟和数据之间的传输延迟
  - 可编程的连续传输延迟
  - 可选择的模式错误检测
  - 最高频率为主时钟

两线接口 TWI

- 只支持主机模式
- 与标准的两线串行存储器兼容

- 从机地址可以为一个字节，两个字节和三个字节
- 连续的读 / 写操作

USART

- 可编程的波特率发生器
- 5 到 9 位的全双工同步或异步串行通讯
  - 1、 1.5 或 2 个停止位，异步模式
  - 1 或 2 个停止位，同步模式
  - 奇偶校验发生器和错误检测
  - 帧错误检测，超速错误检测
  - MSB 先发送或 LSB 先发送
  - 可选的中断 (break) 产生及检测
  - 接收器采样频率为数据频率的 8 或 16 倍
  - 硬件握手信号 RTS - CTS
  - Modem 信号管理 DTR-DSR-DCD-RI， USART1
  - 接收器时间溢出，发送器时间保护 (timeguard)
  - 具有地址产生和检测功能的多地址 (Multi-drop) 通讯模式
- 支持 RS485 通讯，具有驱动器控制信号
- 支持 ISO7816 的 T = 0 或 T = 1 协议，实现与智能卡的接口
  - NACK 信号处理，有限重复次数的错误计数器
- IrDA 调制与解调
  - 通讯速率可达 115.2 Kbps
- 测试模式
  - 远程回环，本地回环，自动回发

串行同步控制器

- 为语音和电信应用提供了串行同步通讯链路
- 包括一个独立的接收器和独立的发送器，以及共用的时钟分频器
- 可配置的帧同步和数据长度
- 接收器和发送器都可以编程为自动启动，或是在检测到帧同步信号上的不同事件之后再启动
- 接收器和发送器包括了数据信号、时钟信号和帧同步信号

定时器 / 计数器

- 3 个 16 位的定时器 / 计数器通道
  - 3 个输出比较或两个输入捕捉
- 如下多种功能：
  - 频率测量
  - 事件计数
  - 时间间隔测量
  - 脉冲产生
  - 延迟
  - PWM
  - 向上递增计数和向下递减计数的能力
- 每个通道都可以进行配置，并包括了：
  - 3 个外部时钟输入

– 5 个外部时钟输入，如 Table 5 所示。

Table 5. 定时器 / 计数器时钟分配

TC 时钟输入	时钟
TIMER_CLOCK1	MCK/2
TIMER_CLOCK2	MCK/8
TIMER_CLOCK3	MCK/32
TIMER_CLOCK4	MCK/128
TIMER_CLOCK5	MCK/1024

– 两个多功能输入 / 输出信号
– 两个作用于所有 3 个 TC 通道的全局寄存器

PWM 控制器

- 4 个通道，每个通道有一个 16 位的计数器
- 公共的时钟发生器，可以提供 13 个不同的时钟
  – 一个 Mod(n) 计数器，提供 11 个时钟
  – 作用于 Mod(n) 计数器输出的两个独立的线性分频器
- 通道的编程是独立的
  – 独立的使能 / 禁止命令
  – 独立的时钟选择
  – 独立的双缓冲周期和占空比
  – 可编程的输出波形极性选择
  – 可编程的输出波形对准方式：中心对准或左边对准

USB 设备端口

- 与 USB V2.0 全速标准兼容，通讯速率为 12 Mbps
- 片内 USB V2.0 全速收发器
- 为端点所使用的片 内 328 字节双口 RAM
- 4 个端点
  – 端点 0: 8 字节
  – 端点 1 和端点 2: 64 字节， ping-pong
  – 端点 3: 64 字节
  – bulk 端点支持 Ping-pong 模式 (两个存储区)
- 挂起 / 继续 (suspend/resume) 逻辑

模数转换器

- 8 通道的 ADC
- 10 位数据，每秒采样次数高达 100K，连续逼近寄存器 ADC
- -2/+2 LSB 积分非线性误差， -1/+2 LSB 差分非线性误差
- 集成的 8-1 多工器，可以有 8 个独立的 3.3V 模拟输入
- 每个通路可以单独使能和禁止
- 对于低电压输入信号可以使用外部电压基准源来提高转换精度
- 多个触发源
  – 硬件或软件触发
  – 外部触发引脚
  – 利用定时器 / 计数器 0-2 的输出 TIOA0-2 进行触发

- 休眠模式和转换序列
  - 触发信号将使芯片自动唤醒：所有使能的通道转换结束之后器件又进入睡眠模式
- 8 个模拟输入中的的 4 个与数字信号共享引脚

ARM7TDMI 处理器综述

概述

ARM7TDMI内核既可以执行32位的ARM®指令集，也可以执行16位的Thumb®指令集，从而可以使用户在高性能和高代码密度之间进行平衡。ARM7TDMI 处理器为冯 - 诺依曼结构，具有三级流水线，即指令获取、解码和执行三个阶段。

ARM7TDMI 处理器的主要特点是：

- ARM7TDMI 基于 ARMv4T 结构
- 两个指令集
  - ARM® 高性能 32 位指令集
  - Thumb® 高代码密度 16 位指令集
- 三级流水线结构
  - 指令获取 (F)
  - 指令解码 (D)
  - 执行 (E)

ARM7TDMI 处理器

有关 ARM7TDMI 更详细的信息请参阅下面的 ARM 文档：

ARM Architecture Reference Manual (DDI 0100E)
ARM7TDMI Technical Reference Manual (DDI 0210B)

指令类型

指令或者是 32 位 (ARM 模式)，或者是 16 位 (THUMB 模式)。

数据类型

ARM7TDMI支持字节(8位) ,半字(16位)以及字(32位)这三种数据类型。字必须与4字节的边界对齐，半字必须与 2 字节的边界对齐。

未对齐的数据访问行为取决于指令类型。

ARM7TDMI 工作模式

ARM7TDMI 基于 ARM 结构 v4T，支持如下 7 种处理器模式：

User：一般的 ARM 程序执行状态

FIQ：设计为高速数据传输或通道处理

IRQ：用于通常的中断处理

Supervisor：用于操作系统的保护模式

Abort mode：实现虚拟内存和 / 或内存保护

System：操作系统的特权用户模式

Undefined：支持硬件协处理器的软件仿真

模式之间的转换可以通过软件进行控制，也可能由中断或例外处理引起。大多数应用程序运行于 User 模式。而非用户模式 ,或特权用户模式 ,则针对中断或例外 ,或者是访问受保护的资源。

ARM7TDMI 寄存器

ARM7TDMI 处理器总共有 37 个寄存器：

- 31 个通用 32 位寄存器
- 6 个状态寄存器

这些寄存器不能够在同一时间都访问到。处理器状态及工作模式决定了哪些寄存器可以被程序员使用。

在任意时刻有 16 个寄存器是可访问的。其他的则与这 16 个寄存器拥有相同的名字，并用于加速例外的处理。

寄存器 R15 为程序计数器 (PC)，可以用于所有的指令来获取相对于当前指令的数据。

R14 保留着子程序调用的返回地址。

R13 通常用做堆栈指针。

.

Table 6. ARM7TDMI ARM 模式及寄存器规划

User 和 System 模式	Supervisor模式	Abort 模式	Undefined 模式	IRQ 模式	FIQ 模式
R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3

Table 6. ARM7TDMI ARM 模式及寄存器规划

User 和 System 模式	Supervisor模式	Abort 模式	Undefined 模式	IRQ 模式	FIQ 模式
R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8_FIQ
R9	R9	R9	R9	R9	R9_FIQ
R10	R10	R10	R10	R10	R10_FIQ
R11	R11	R11	R11	R11	R11_FIQ
R12	R12	R12	R12	R12	R12_FIQ
R13	R13_SVC	R13_ABORT	R13_UNDEF	R13_IRQ	R13_FIQ
R14	R14_SVC	R14_ABORT	R14_UNDEF	R14_IRQ	R14_FIQ
PC	PC	PC	PC	PC	PC
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_SVC	SPSR_ABORT	SPSR_UNDEF	SPSR_IRQ	SPSR_FIQ

与模式相关的分区的寄存器

寄存器 R0 到 R7 是不分区的。也就是说，在所有的工作模式下，它们指向的是相同的 32 位物理寄存器。它们也是通用寄存器，与处理器结构没有直接的关系，可以出现在任何指令允许它们出现的地方。

R8 到 R14 则是分区的寄存器。它们指向的物理寄存器根据处理器工作模式的不同而不同。

模式和例外处理

所有的例外都有分区的寄存器 R14 和 R13。

例外发生后， R14 保存了例外处理的返回地址。例外处理结束后，通过这个地址程序返回到例外发生时的指令。

对于各种例外模式， R13 也是分区的，从而可以外各个例外处理程序提供单独的堆栈指针。

快速中断模式还实现了 R8 到 R12 的分区，从而中断处理程序可以不保存这些寄存器就开始执行中断处理。

第 7 个处理模式 ,即 System 模式 ,没有任何分区的寄存器。它使用与 User 模式相同的寄存器。在 System 模式里可以运行那些需要特权模式的任务，并允许它们引发各种级别的例外。

状态寄存器

所有其他的处理器状态保存于状态寄存器。当前处理器状态保存于当前程序状态寄存器 (Current Program Status Register,CPSR)。其内容为：

- 4 个 ALU 标志 (负，零，进位，溢出)
- 两个中断禁止位 (每种中断一个)
- 一个比特用来指示 ARM 状态还是 Thumb 状态
- 5 个比特用来解码当前的处理器模式

所有 5 个例外模式还拥有一个保存的程序状态寄存器 (Saved Program Status Register,SPSR)，这个寄存器保存了在例外发生之前的那个任务的 CPSR。

例外类型

ARM7TDMI 支持 5 种类型的例外，每一种都有对应的特权处理模式。这些例外类型是：

- 快速中断 (FIQ)
- 普通中断 (IRQ)
- 内存异常中断 (用来实现内存保护或虚拟内存)
- 尝试执行未定义的指令
- 软件中断 (SWI)

例外由片内外的中断源产生。

在同一时间可以有多个例外发生。

例外发生时，此例外对应的、分区的 R14 和 SPSR 将当前状态保存下来。

从例外返回时需要将 SPSR 拷贝回 CPSR , 同时将 R14 的值赋给 PC。实现的方法有如下两个 :

- 使用数据处理指令， S 置位， PC 为目的寄存器
- 使用 LDM 指令

ARM 指令集概览

ARM 指令集分为：

- 分支跳转指令
- 数据处理指令
- 状态寄存器传输指令
- 加载和保存指令
- 协处理器指令
- 例外产生指令

ARM 指令可以有条件地执行。每个指令都包括 4 比特的条件代码域 (bit[31:28])。

Table 7 给出了 ARM 指令表。

Table 7. ARM 指令表

助记符	操作	助记符	操作
MOV	Move	CDP	Coprocessor Data Processing
ADD	Add	MVN	Move Not
SUB	Subtract	ADC	Add with Carry
RSB	Reverse Subtract	SBC	Subtract with Carry
CMP	Compare	RSC	Reverse Subtract with Carry
TST	Test	CMN	Compare Negated
AND	Logical AND	TEQ	Test Equivalence
EOR	Logical Exclusive OR	BIC	Bit Clear
MUL	Multiply	ORR	Logical (inclusive) OR
SMULL	Sign Long Multiply	MLA	Multiply Accumulate
SMLAL	Signed Long Multiply Accumulate	UMULL	Unsigned Long Multiply
MSR	Move to Status Register	UMLAL	Unsigned Long Multiply Accumulate
B	Branch	MRS	Move From Status Register
BX	Branch and Exchange	BL	Branch and Link

Table 7. ARM 指令表

助记符	操作
LDR	Load Word
LDRSH	Load Signed Halfword
LDRSB	Load Signed Byte
LDRH	Load Half Word
LDRB	Load Byte
LDRBT	Load Register Byte with Translation
LDRT	Load Register with Translation
LDM	Load Multiple
SWP	Swap Word
MCR	Move To Coprocessor
LDC	Load To Coprocessor

助记符	操作
SWI	Software Interrupt
STR	Store Word
STRH	Store Half Word
STRB	Store Byte
STRBT	Store Register Byte with Translation
STRT	Store Register with Translation
STM	Store Multiple
SWPB	Swap Byte
MRC	Move From Coprocessor
STC	Store From Coprocessor

Thumb 指令集概览

Thumb 指令集是 ARM 指令集经过重新编码的子集。

Thumb 指令集分为：

- 分支跳转指令
- 数据处理指令
- 加载和保存指令
- 加载和保存 (多个数据) 指令
- 例外产生指令

在 Thumb 模式中 8 个通用寄存器，R0 到 R7，是可访问的。还可以访问的是程序计数器 (ARM 寄存器 15)，链接寄存器 (ARM 寄存器 14) 以及堆栈指针 (ARM 寄存器 13)。还有一些指令对 ARM 寄存器 R8 到 R15 有有限的访问权限。

Table 8 给出了 Thumb 指令集。

Table 8. Thumb 指令集

助记符	操作
MOV	Move
ADD	Add
SUB	Subtract
CMP	Compare
TST	Test
AND	Logical AND
EOR	Logical Exclusive OR
LSL	Logical Shift Left
ASR	Arithmetic Shift Right
MUL	Multiply
B	Branch
BX	Branch and Exchange
LDR	Load Word

助记符	操作
MVN	Move Not
ADC	Add with Carry
SBC	Subtract with Carry
CMN	Compare Negated
NEG	Negate
BIC	Bit Clear
ORR	Logical (inclusive) OR
LSR	Logical Shift Right
ROR	Rotate Right
BL	Branch and Link
SWI	Software Interrupt
STR	Store Word

Table 8. Thumb 指令集

助记符	操作	助记符	操作
LDRH	Load Half Word	STRH	Store Half Word
LDRB	Load Byte	STRB	Store Byte
LDRSH	Load Signed Halfword	LDRSB	Load Signed Byte
LDMIA	Load Multiple	STMIA	Store Multiple
PUSH	Push Register to stack	POP	Pop Register from stack

AT91SAM7S64 调试及测试特点

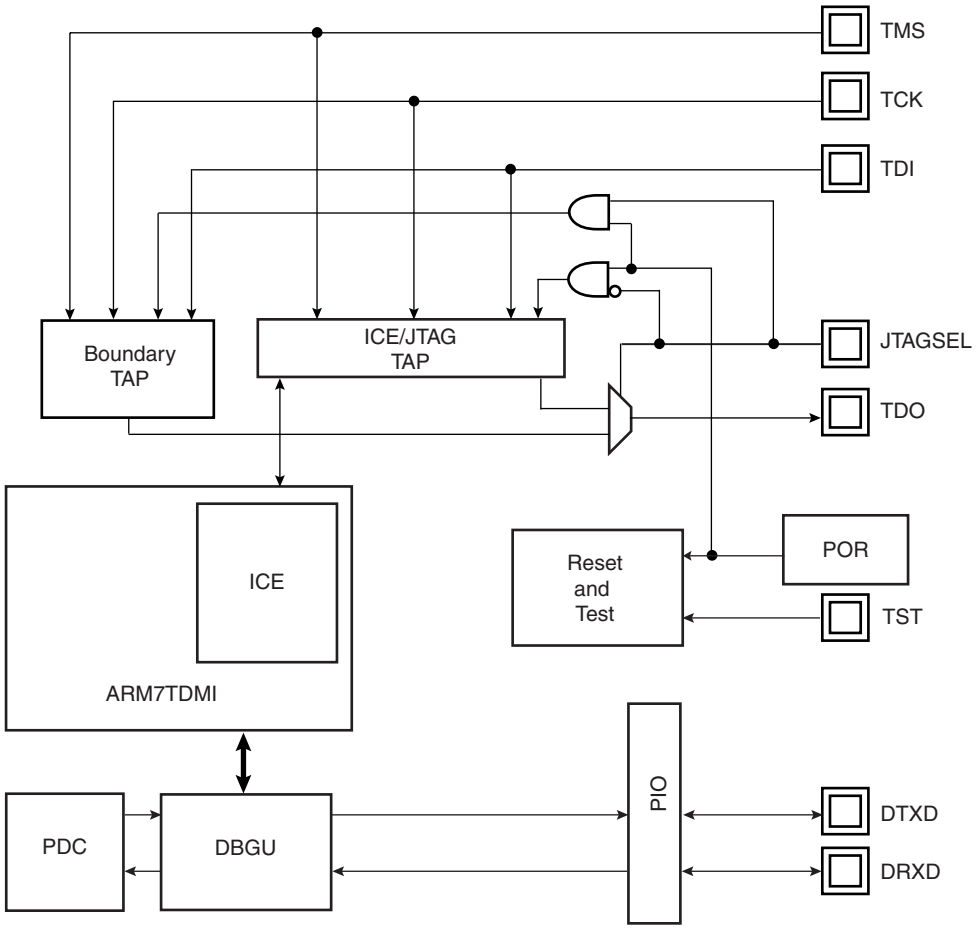
说明

AT91SAM7S64 具有调试及测试能力。 JTAG/ICE (In-Circuit Emulator) 用于标准的调试功能，如下载代码，在程序里实现单步执行等。调试单元提供了一个两线 UART，可以利用它将应用程序上载到片内 SRAM。它管理着 COMMTX 和 COMMRX 信号的中断处理，实现对调试通讯通道活动的跟踪。

通过一整套复杂的调试与测试输入 / 输出引脚， PC 端的测试环境可以直接访问这些调试 / 测试功能。

方框图

Figure 10. 调试及测试框图

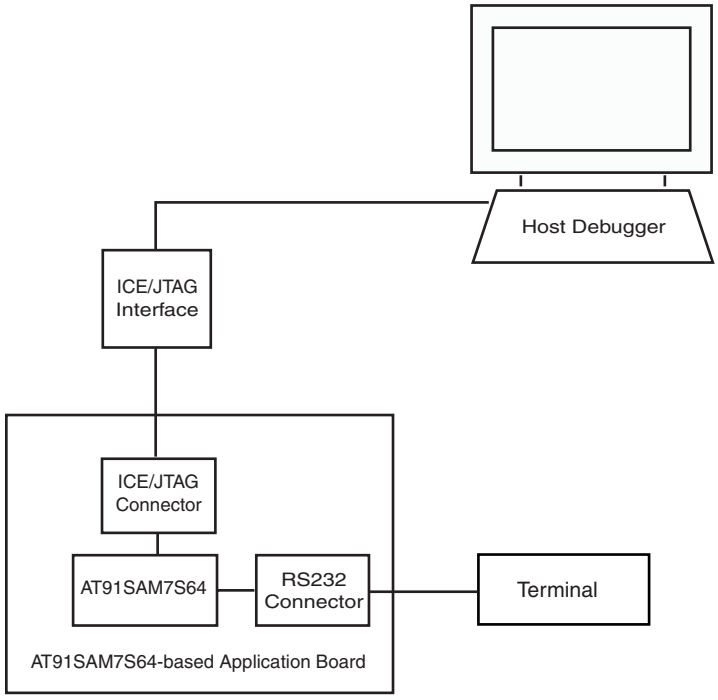


应用例子

调试环境

Figure 11 on page 38 展示了一个完整的调试环境例子。ICE/JTAG 接口用于标准的调试功能，如下载代码和单步执行程序。

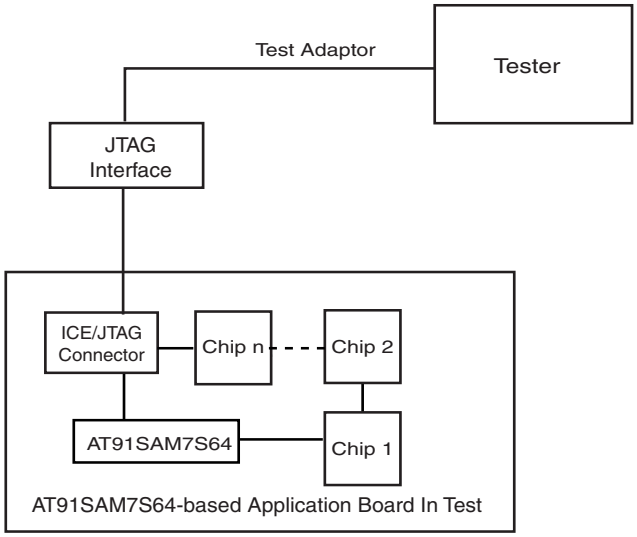
Figure 11. 调试环境例子



测试环境

Figure 12 on page 39 展示了测试环境的例子。测试向量由测试仪发送和解释。在这个例子中，接受测试的板子上有一些与 JTAG 兼容的器件。这些器件可以连接到一起组成一个扫描链。

Figure 12. 测试环境例子



调试及测试引脚说明

Table 9. 调试及测试引脚列表

引脚名称	功能	类型	有效电平
复位 / 测试			
NRST	处理器复位	输入 / 输出	低
TST	测试模式选择	输入	高
ICE 及 JTAG			
TCK	测试时钟	输入	
TDI	测试数据输入	输入	
TDO	测试数据输出	输出	
TMS	测试模式选择	输入	
JTAGSEL	JTAG 选择	输入	
调试单元			
DRXD	调试接收数据	输入	
DTXD	调试发送数据	输出	

功能描述

测试引脚

TST引脚用来定义芯片工作模式。在正常工作模式下用户必须保证这个引脚接到低电平。高电平则用于生产测试。

嵌入式在线仿真器 - ICE

ARM7TDMI 嵌入式在线仿真是通过 ICE/JTAG 得到支持的。 ARM7TDMI 的内部状态通过 ICE/JTAG 得以检查。

ARM7TDMI 处理器具有实现先进的调试特性的硬件扩展：

- 在暂停 (halt) 模式下，“ 存储多个数据” (STM) 可以插入到指令流水线。这个指令可以输出 ARM7TDMI 寄存器的内容。然后这些数据以串行的方式从 TDI 输出 ，而不影响系统的其他部分。
- 在监控模式下， JTAG 接口用于在调试器和运行于 ARM7TDMI 处理器内部的简单监控程序之间传输数据。

ARM7TDMI处理器有三个扫描链分别支持测试、调试及编程。所有扫描链都由ICE/JTAG控制。

JTAGSEL引脚为低时选择嵌入式ICE模式。不能在ICE和JTAG模式之间直接进行切换。JTAG-SEL 变化后必须进行芯片复位。

进一步的嵌入式ICE信息请参阅ARM7TDMI (Rev4) Technical Reference Manual (DDI0210B)。

调试单元

调试单元提供了两线 (DXRD 和 TXRD) USART 用于一些调试及跟踪的目的。同时这也是现场编程和调试监控通讯的理想方法。此外，由于两个外设数据控制器通道的存在，处理器处理数据包的时间可以减至最小。

调试单元还管理着 COMMTX 和 COMMRX 的中断处理。这是两个来自 ICE 的信号，它们跟踪着调试通讯通道的活动。调试单元可以阻止通过 ICE 接口对系统进行的访问。

调试单元可以用来上载应用程序到片内 SRAM。当引导程序没有发现有效的应用程序时即激活这个功能。用来加载应用程序的协议是 XMODEM。

调试单元芯片 ID 寄存器保存了有关产品版本及内部配置的信息。

AT91SAM7S64 的调试单元芯片 ID 的数值为 0x27090540。

有关调试单元的更多信息请参阅 “Debug Unit (DBGU)” on page 179。

IEEE 1149.1 JTAG 边界扫描

通过 IEEE 1149.1 JTAG 边界扫描可以访问每个引脚，而不必关心器件的封装形式。

JTAGSEL 为高时使能 IEEE 1149.1 JTAG 边界扫描。实现的功能有 SAMPLE、 EXTEST 和 BYPASS。在 ICE 调试模式下 ,ARM 处理器将返回一个非 JTAG 芯片 ID 到 ICE 系统。这与 IEEE 1149.1 JTAG 是不兼容的。

JTAG 模式和 ICE 模式不能直接进行切换。 JTAGSEL 电平改变之后必须对芯片进行复位。

Atmel 提供了边界扫描叙词语言 (BSDL) 文件以方便客户设置测试系统。

JTAG 边界扫描寄存器

边界扫描寄存器 (BSR) 包含与各个引脚和相关控制信号对应的 96 个比特。

每个 AT91SAM7S64 输入 / 输出引脚在 BSR 里都占 3 个比特。OUTPUT 表示数据可以强制输出到引脚；INPUT 用来观察加载于引脚的数据；CONTROL 则控制引脚的方向。

Table 10. AT91SAM7S64 JTAG 边界扫描寄存器

比特流的顺序号	引脚名称	引脚类型	相关的 BSR 单元
96	PA17/PGMD5/AD0	IN/OUT	INPUT
95			OUTPUT
94			CONTROL
93	PA18/PGMD6/AD1	IN/OUT	INPUT
92			OUTPUT
91			CONTROL
90	PA21/PGMD9	IN/OUT	INPUT
89			OUTPUT
88			CONTROL
87	PA19/PGMD7/AD2	IN/OUT	INPUT
86			OUTPUT
85			CONTROL
84	PA20/PGMD8/AD3	IN/OUT	INPUT
83			OUTPUT
82			CONTROL
81	PA16/PGMD4	IN/OUT	INPUT
80			OUTPUT
79			CONTROL
78	PA15/PGM3	IN/OUT	INPUT
77			OUTPUT
76			CONTROL
75	PA14/PGMD2	IN/OUT	INPUT
74			OUTPUT
73			CONTROL
72	PA13/PGMD1	IN/OUT	INPUT
71			OUTPUT
70			CONTROL
69	PA22/PGMD10	IN/OUT	INPUT
68			OUTPUT
67			CONTROL
66	PA23/PGMD11	IN/OUT	INPUT
65			OUTPUT
64			CONTROL

Table 10. AT91SAM7S64 JTAG 边界扫描寄存器

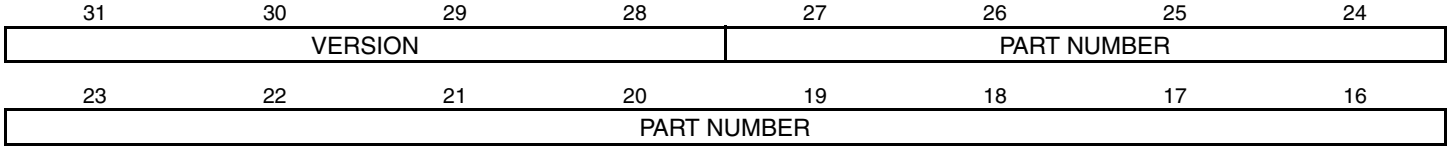
比特流的顺序号	引脚名称	引脚类型	相关的 BSR 单元
63	PA24/PGMD12	IN/OUT	INPUT
62			OUTPUT
61			CONTROL
60	PA12/PGMD0	IN/OUT	INPUT
59			OUTPUT
58			CONTROL
57	PA11/PGMM3	IN/OUT	INPUT
56			OUTPUT
55			CONTROL
54	PA10/PGMM2	IN/OUT	INPUT
53			OUTPUT
52			CONTROL
51	PA9/PGMM1	IN/OUT	INPUT
50			OUTPUT
49			CONTROL
48	PA8/PGMM0	IN/OUT	INPUT
47			OUTPUT
46			CONTROL
45	PA7/PGMNVALID	IN/OUT	INPUT
44			OUTPUT
43			CONTROL
42	PA6/PGMNOE	IN/OUT	INPUT
41			OUTPUT
40			CONTROL
39	PA5/PGMRDY	IN/OUT	INPUT
38			OUTPUT
37			CONTROL
36	PA4/PGMNCMD	IN/OUT	INPUT
35			OUTPUT
34			CONTROL
33	PA25/PGMD13	IN/OUT	INPUT
32			OUTPUT
31			CONTROL
30	PA26/PGMD14	IN/OUT	INPUT
29			OUTPUT
28			CONTROL

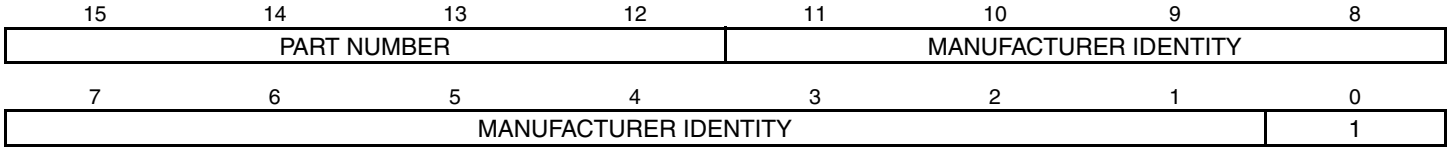
Table 10. AT91SAM7S64 JTAG 边界扫描寄存器

比特流的顺序号	引脚名称	引脚类型	相关的 BSR 单元
27	PA27/PGMD15	IN/OUT	INPUT
26			OUTPUT
25			CONTROL
24	PA28	IN/OUT	INPUT
23			OUTPUT
22			CONTROL
21	PA3	IN/OUT	INPUT
20			OUTPUT
19			CONTROL
18	PA2	IN/OUT	INPUT
17			OUTPUT
16			CONTROL
15	PA1/PGMEN1	IN/OUT	INPUT
14			OUTPUT
13			CONTROL
12	PA0/PGMEN0	IN/OUT	INPUT
11			OUTPUT
10			CONTROL
9	PA29	IN/OUT	INPUT
8			OUTPUT
7			CONTROL
6	PA30	IN/OUT	INPUT
5			OUTPUT
4			CONTROL
3	PA31	IN/OUT	INPUT
2			OUTPUT
1			CONTROL
0	ERASE	IN	INPUT

ID 代码寄存器

访问类型：只读





VERSION[31:28]: 芯片版本号

0x1

PART NUMBER[27:12]: 芯片型号

0x5B06

MANUFACTURER IDENTITY[11:1]

0x01F

Bit[0] Required by IEEE Std. 1149.1.

0x1

JTAG ID 代码为 05B0_603F

复位控制器 (RSTC)

概述

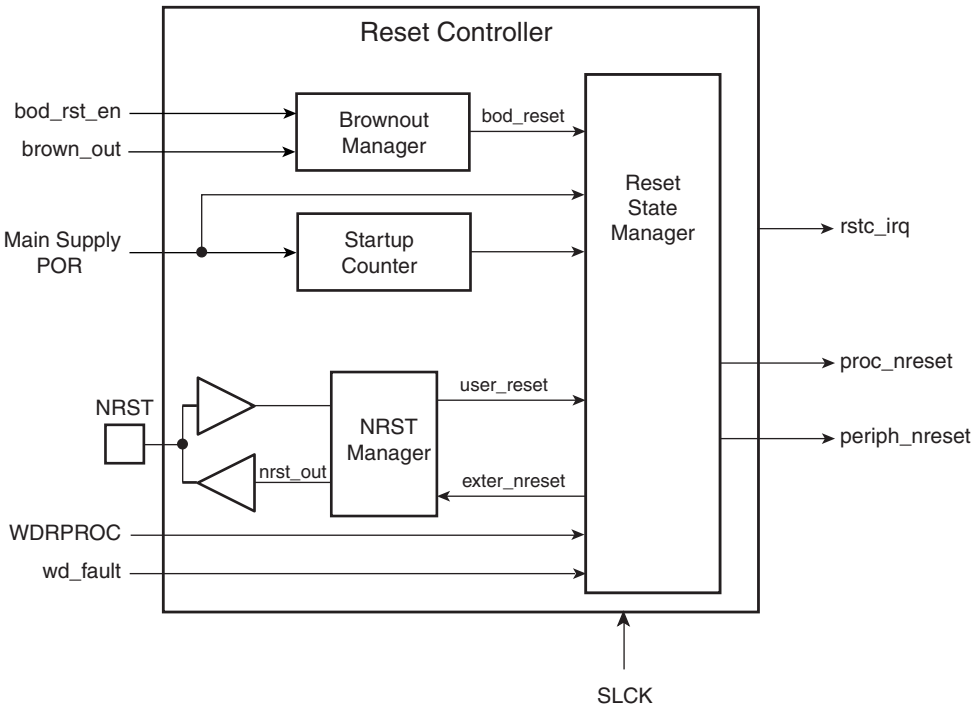
基于上电复位单元的复位控制器 (RSTC) 处理系统的所有复位 , 而无需其它器件。它可以给出上一次复位源的信息。

复位控制器可以独立地、或同时驱动外部复位和外设及处理器复位。

掉电检测可以防止处理器进入不可预测的状态。

方框图

Figure 13. 复位控制器框图



功能说明

复位控制器由 NRST 管理器、掉电检测管理器、启动计数器和复位状态管理器组成。它运行于慢速时钟，可以产生如下复位信号：

- proc_nreset：处理器复位。它同时也复位看门狗定时器。
- periph_nreset：作用于所有的外设。
- nrst_out：驱动 NRST 引脚。

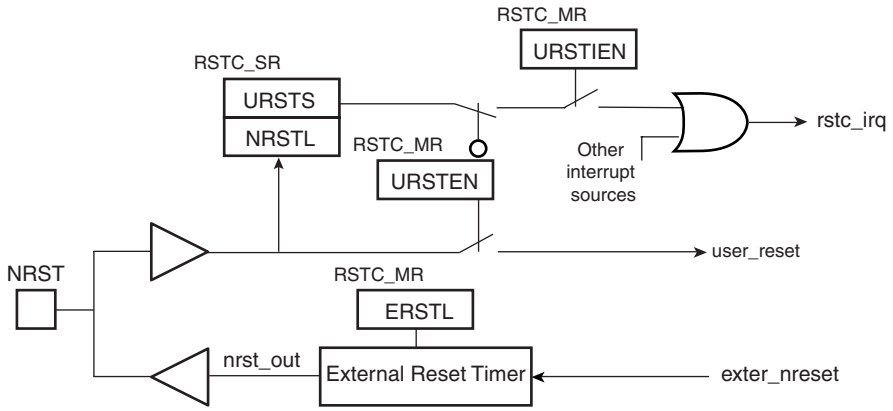
这些复位信号由复位控制器或者基于外部事件，或者基于软件行为产生。复位状态管理器控制着这些信号的产生，并在需要激活 NRST 引脚的时候为 NRST 管理器提供信号。

NRST 管理器控制 NRST 信号保持预先编好程的一段时间，从而控制外部器件的复位。

NRST 管理器

NRST 管理器对 NRST 引脚的输入进行采样，并在复位状态管理器需要的时候将引脚电平拉低。Figure 14 给出了 NRST 管理器的方框图。

Figure 14. NRST 管理器



NRST 信号或中断

NRST 管理器以低速时钟对 NRST 引脚信号进行采样。当检测到信号为低时，用户复位的信号将报告给复位状态管理器。

此外，通过编程还可以使 NRST 管理器在 NRST 为低时并不触发复位。这可以通过将 RSTC_MR 寄存器的 URSTEN 位清零来实现。

引脚 NRST 的电平可以在任何时候通过读取寄存器 RSTC_SR 的 NRSTL 来了解。一旦 NRST 上施加了有效信号，寄存器 RSTC_SR 的位 URSTS 置位。只有读取 RSTC_SR 之后这一位才清零。

通过编程还可以使复位控制器产生中断，而不是复位。方法就是置位 RSTC_MR 寄存器的 URSTIEN。

NRST 外部复位控制

复位状态管理器产生 ext_nreset 信号来拉低 NRST 引脚。同时，“nrst_out” 信号被 NRST 管理器拉低，并持续由 RSTC_MR 寄存器的 ERSTL 域控制的一段时间。这段名为 EXTERNAL_RESET_LENGTH 的时间持续 $2^{(ERSTL+1)}$ 个慢速时钟周期。所以其时间范围为 60 µs 到 2 秒。 ERSTL 为 0 时 NRST 脉冲持续两个时钟周期。

这个特性使得复位控制器可以塑造 NRST 引脚的电平，从而保证 NRST 低电平持续时间满足连接到系统复位的外部器件的复位要求。

掉电检测管理器

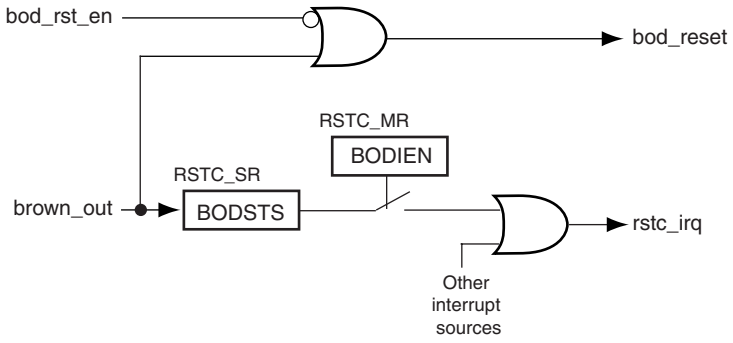
掉电检测可以防止处理器在电源下降到某个特定电平之后进入不可预测的状态。当 VDDCORE 低于掉电检测门限时，掉电检测管理器通过激活 bod_reset 信号来请求掉电检测复位。

程序员可以通过拉低 bod_rst_en 信号的方法来禁止掉电检测复位，即锁定 Flash 中对应的通用 NVM 位。然后掉电检测复位就不会再发生了。此时可以通过 寄存器RSTC_SR的位BODSTS来查看是否发生了掉电检测。 BODSTS 置位后只能通过读取 RSTC_SR 来清除。

若寄存器 RSTC_MR 的 BODIEN 置位，则 BODSTS 可以触发中断。

芯片出厂时掉电检测复位是禁止的。

Figure 15. 掉电检测管理器



复位状态

复位状态管理器处理不同的复位源，并产生内部复位信号。它通过状态寄存器 RSTC_SR 的 RSTTYP 域来报告复位状态。处理器复位释放后 RSTTYP 即得以更新。

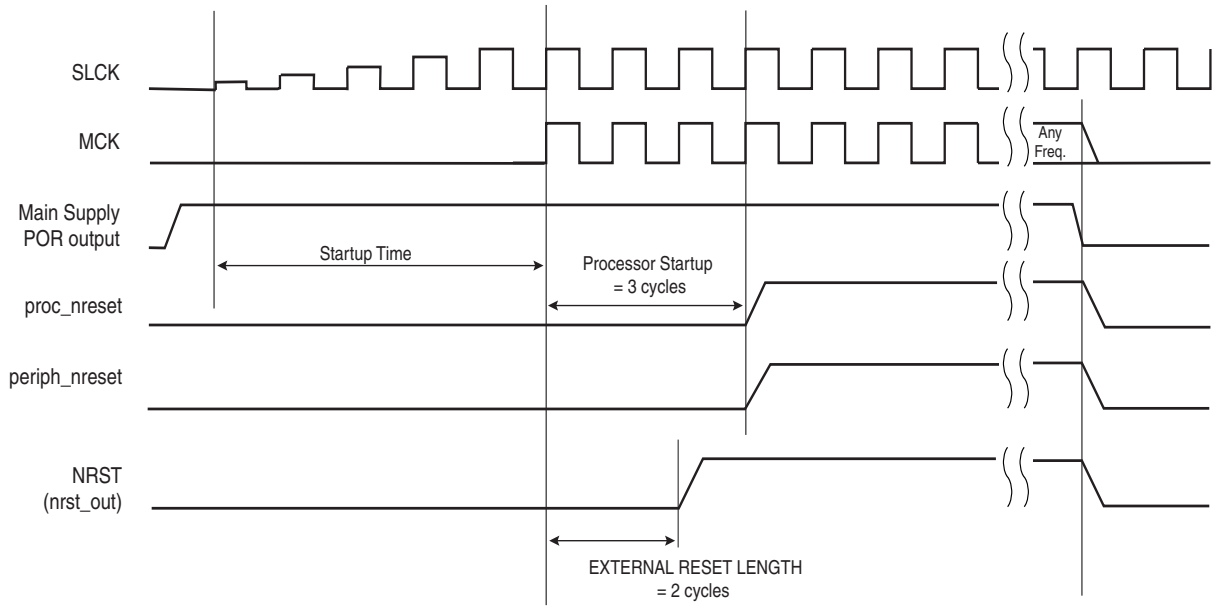
上电复位

VDDCORE 上电后 ,主电源 POR 单元输出被运行于慢速时钟的启动计数器所过滤。这个计数器的目的是保证启动芯片之前慢速时钟振荡器先稳定下来。

Figure 16 所示的启动时间是硬件决定的，并与慢速时钟振荡器的启动时间兼容。经过启动时间之后，复位信号被释放， RSTC_SR 寄存器的 RSTTYP 得以更新，指明发生了上电复位。

当主电源 POR 单元检测到 VDDCORE 为低时，所有的复位信号立即生效。

Figure 16. 上电复位



用户复位

当 NRST 引脚电平为低，且寄存器 RSTC_MR 的 URSTEN 为 1 时即进入用户复位。 NRST 输入信号被同步到 SLCK 以保证系统的正确运行。

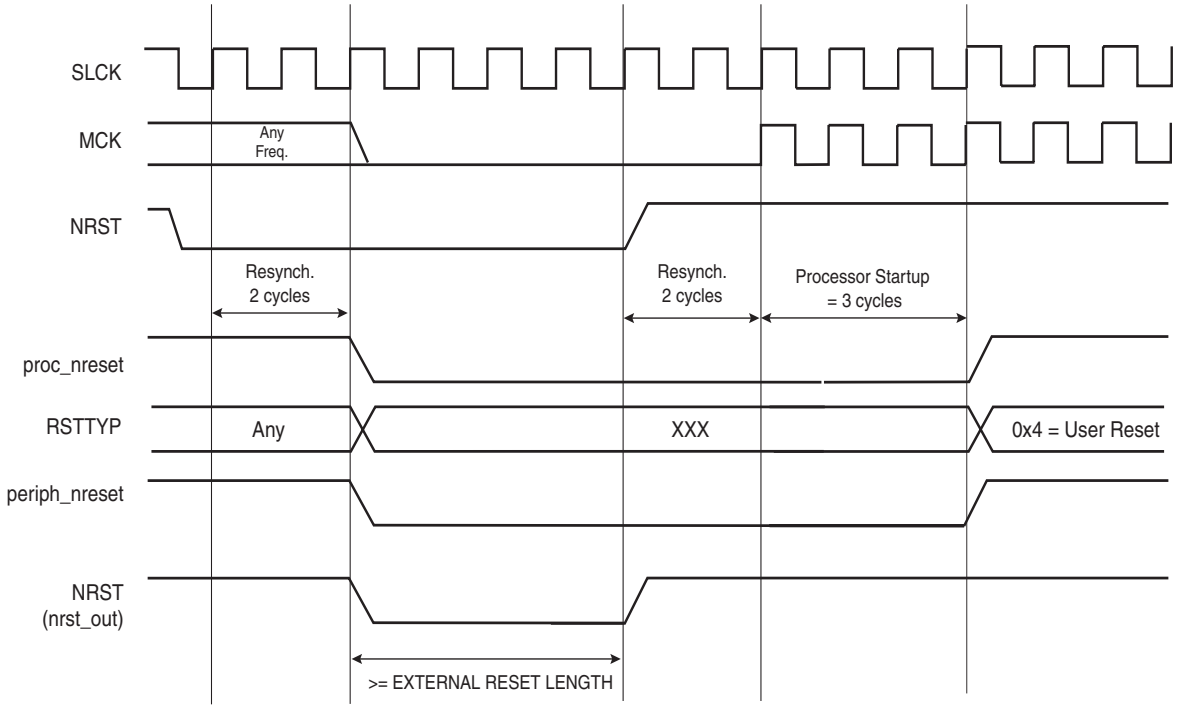
一旦检测到 NRST 为低电平，系统即进入用户复位。同时还引发处理器复位和外设复位。

NRST 电平拉高后，经过两个周期的重新同步时间和三个周期的处理器启动时间，处理器即退出用户复位。一旦 NRST 为高，处理器时钟即重新使能。

处理器复位信号释放之后，状态寄存器的 RSTTYP 域即更新为 0x4，表示发生了用户复位。

NRST 管理器保证 NRST 信号持续 EXTERNAL_RESET_LENGTH 个慢速时钟周期，正如域 ERSTL 编程的那样。然而，如果由于被外部电路拉低而使得 NRST 并没有在预定时间变高，内部复位将保持有效直到 NRST 变高。

Figure 17. 用户复位状态



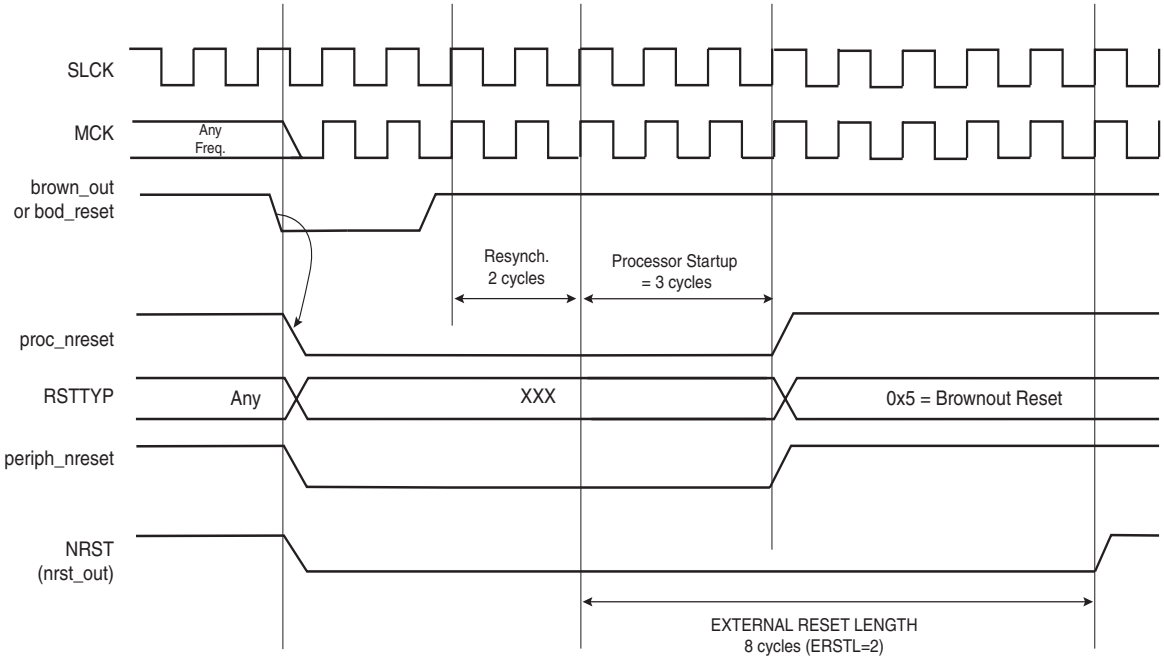
掉电检测复位

brown_out/bod_reset 信号有效时，复位状态管理器立即进入掉电检测复位。处理器复位、外设复位和外部复位信号同时生效。

brown_out/bod_reset 变高后，经过两个周期的重新同步时间和额外的 3 个慢速时钟周期，处理器退出掉电检测复位。同时外部复位被触发。

处理器复位释放之后， RSTC_SR 寄存器的 RSTTYP 更新为 0x5，表明发生了掉电检测复位。

Figure 18. 掉电检测复位状态



软件复位

复位控制器提供了几个命令来激活不同的复位信号。执行这些命令只需要将控制寄存器 RSTC_CR 的如下控制位写 1：

- PROCRST：置 1 将复位处理器和看门狗
- PERRST：置 1 将复位所有的外设，包括存储器系统，特别是重映像 (Remap) 命令。外设复位一般用于调试的目的。
- EXTRST：置 1 将拉低 NRST 引脚，并保持由模式寄存器 RSTC_MR 的 ERSTL 域定义的一段时间。

一旦软件设置了上述的某一位或某几位，处理器即进入软件复位。所有这些命令可以单独执行，也可以同时执行。软件复位持续 3 个慢速时钟周期。

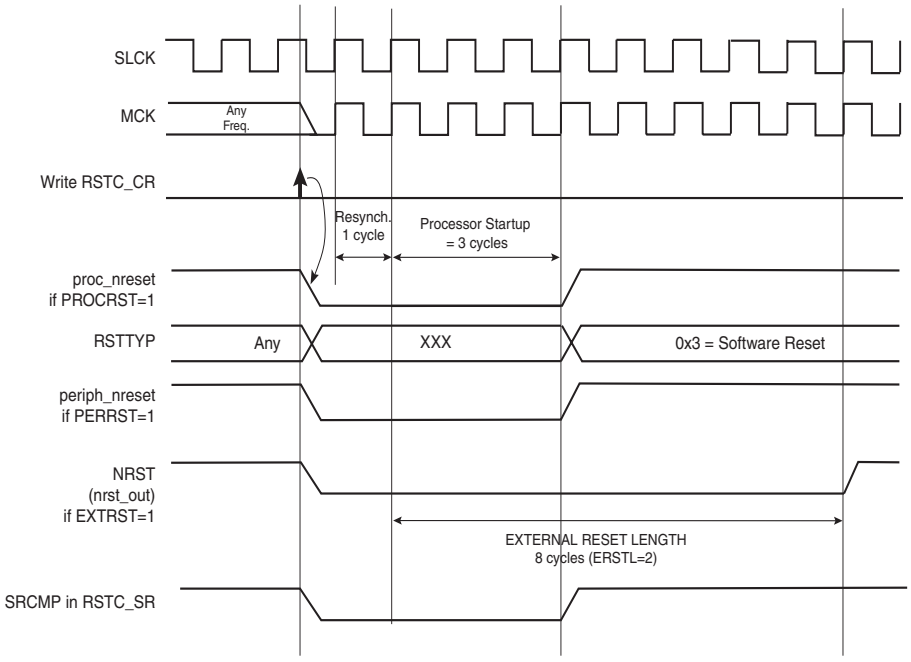
一旦执行了控制寄存器写操作，内部各个复位信号立即生产。这可以通过主时钟 (MCK) 检测出来：处理器退出软件复位时这些复位信号即被释放，亦即它们都同步于 SLCK。

如果 EXTRST 置 1， nrst_out 信号是否有效还取决于 ERSTL 的设置。然而 NRST 的下降沿并不会导致用户复位。

若仅有 PROCRST 置位，复位控制器将通过状态寄存器 RSTC_SR 的 RSTTYP 域报告软件复位。 RSTTYP 不报告其它的软件复位。

一旦发生了软件复位，状态寄存器 RSTC_SR 的位 SRCMP (Software Reset Command in Progress- 正在执行软件复位命令) 即置位。处理器退出软件复位后它即被清零。SRCMP 置位后无法执行其它软件复位，对 RSTC_CR 的写操作也没有任何效果。

Figure 19. 软件复位



看门狗复位

看门狗错误发生时引发看门狗复位。这个状态将持续 3 个慢速时钟周期。

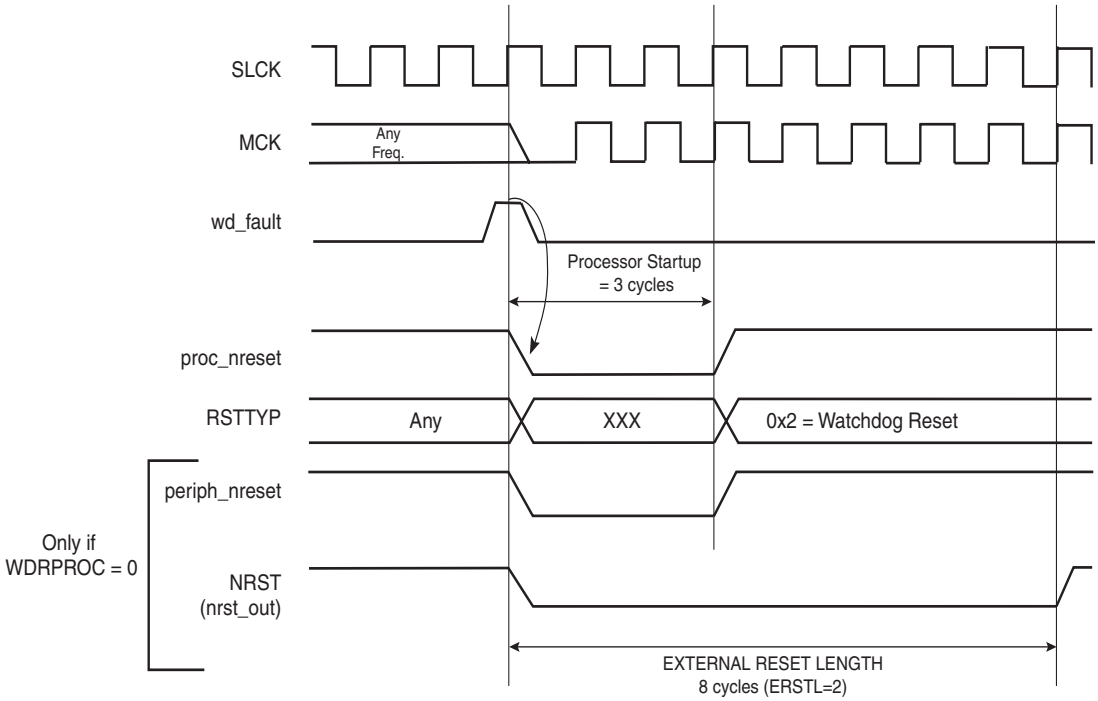
看门狗复位时，内部各个复位信号的产生取决于 WDT_MR 寄存器的位 WDRPROC：

- 若 WDRPROC 为 0，处理器复位和外设复位信号有效。NRST 引脚也被拉低，持续时间与 ERSTL 有关。但是 NRST 引脚的低电平并不会引起用户复位。
- 若 WDRPROC = 1，则只有处理器复位有效。

看门狗定时器由 proc_nreset 信号复位。如果 WDRSTEN 置位，看门狗溢出总是会引起处理器复位，因此看门狗定时器总是在看门狗复位之后复位。在缺省条件下看门狗是使能的，而且溢出时间设置为最长。

寄存器WDT_MR的WDRSTEN清零时，看门狗错误(定时器溢出)对复位控制器没有任何影响。

Figure 20. 看门狗复位



复位状态优先级

复位状态管理器管理的复位源优先级以递减的方式排列如下：

- 上电复位
- 掉电检测复位
- 看门狗复位
- 软件复位
- 用户复位

特例则排列如下：

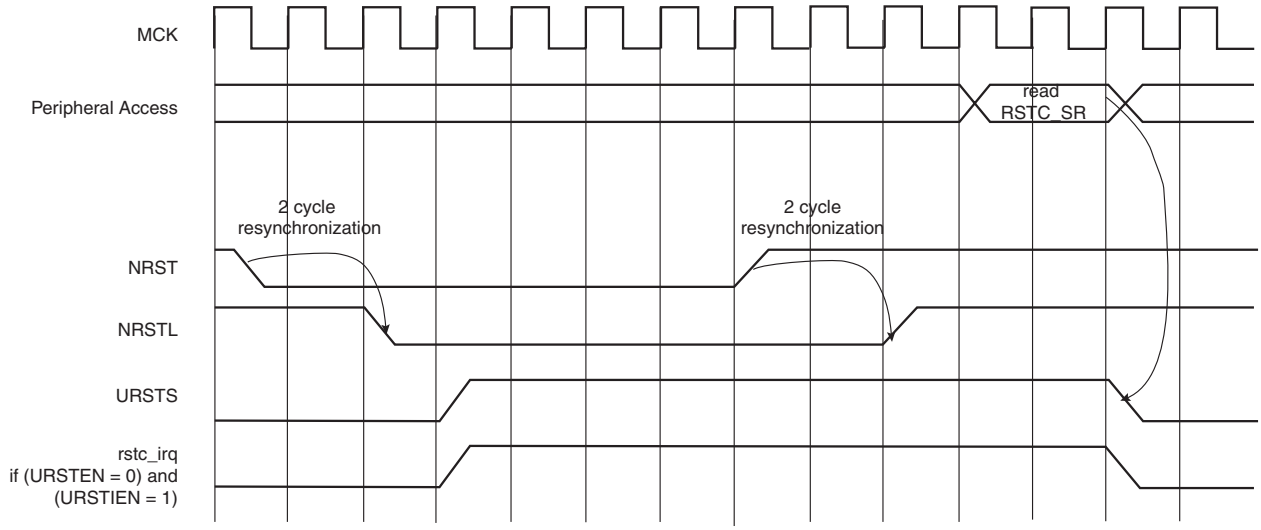
- 用户复位状态：
  - 看门狗事件永远不会发生，因为看门狗定时器被 proc_nreset 信号复位了
  - 软件复位不会发生，因为处理器复位已经被激活了
- 软件复位状态：
  - 看门狗事件优先级高于当前状态
  - NRST 没有效用
- 看门狗复位状态：
  - 处理器复位已经激活，所以不可能进行软件复位的编程
  - 不能进入用户复位

复位控制器状态寄存器

复位控制器状态寄存器 (RSTC_SR) 给出了如下几个状态域：

- RSTTYP：表明了上一次复位的复位源
- SRCMP：表明软件复位命令正在执行当中，不能够执行其它的软件复位。当前软件复位结束后这一位自动清零。
- NRSTL：给出了在每一个主时钟 MCK 上升沿采样到的 NRST 引脚的电平
- URSTS：NRST 信号从高电平到低电平的变化将置位 RSTC_SR 寄存器的 URSTS。这个电平变换也是在主时钟 MCK 的上升沿采样到的 (参见 Figure 21)。如果用户复位已经被禁止 (URSTEN = 0) ,而且通过 RSTC_MR 寄存器的 URSTIEN 使能了中断 ,则 URSTS 将触发中断。读取状态寄存器 RSTC_SR 将复位 URSTS 并清除中断。
- BODSTS：当掉电检测复位被禁止时 (bod_rst_en = 0) 这一位表示发生了掉电检测。如果通过 RSTC_MR 寄存器的 URSTIEN 使能了中断则 BODSTS 将触发中断。读取状态寄存器 RSTC_SR 将复位 BODSTS 并清除中断。

Figure 21. 复位控制器状态和中断



复位控制器 (RSTC) 用户接口

Table 11. 复位控制器寄存器

地址偏移	寄存器	名称	访问类型	复位值
0x00	控制寄存器	RSTC_CR	只写	-
0x04	状态寄存器	RSTC_SR	只读	0x0000_0000
0x08	模式寄存器	RSTC_MR	读 / 写	0x0000_0000

复位控制器控制寄存器

寄存器名称： RSTC_CR

访问类型： 只写

31	30	29	28	27	26	25	24
KEY							
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–		–
7	6	5	4	3	2	1	0
–	–	–	–	EXTRST	PERRST	–	PROCRST

- **PROCRST: 处理器复位**

0 = 无效。

1 = 若 KEY 是正确的，则复位处理器。

- **PERRST: 外设复位**

0 = 无效。

1 = 若 KEY 是正确的，则复位所有外设。

- **EXTRST: 外部复位**

0 = 无效。

1 = 若 KEY 是正确的，则拉低 NRST 引脚。

- **KEY: 预设值**

0xA5。写入其它任何数值都将使本次写操作失败。

复位控制器状态寄存器

寄存器名称： RSTC_SR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	SRCMP	NRSTL

15	14	13	12	11	10	9	8
–	–	–	–	–	RSTTYP		

7	6	5	4	3	2	1	0
–	–	–	–	–	–	BODSTS	URSTS

- **URSTS: 用户复位状态**

0 = 自上次读取 RSTC_SR 以来 NRST 引脚上没有发生从高电平到低电平的转换。

1 = 自上次读取 RSTC_SR 以来 NRST 引脚上至少检测到一次从高电平到低电平的转换。

- **BODSTS: 掉电检测状态**

0 = 自上次读取 RSTC_SR 以来没有发生掉电检测。

1 = 自上次读取 RSTC_SR 以来已经发生过掉电检测。

- **RSTTYP: 复位类型**

报告上一次处理器复位的原因。读取 RSTC_SR 不会影响它的数值。

RSTTYP			复位类型	说明
0	0	0	上电复位	VDDCORE 上升
0	1	0	看门狗复位	发生了看门狗错误 (定时器溢出)
0	1	1	软件复位	软件要求的处理器复位
1	0	0	用户复位	NRST 引脚检测为低
1	0	1	掉电检测复位	发生了掉电检测复位

- **NRSTL: NRST 引脚电平**

记录了主时钟 MCK 上升沿时刻的 NRST 引脚电平。

- **SRCMP: 软件复位命令执行中**

0 = 复位控制器没有在执行软件复位命令。复位控制器可以执行软件复位命令了。

1 = 复位控制器正在执行软件复位命令。复位控制器忙。

复位控制器模式寄存器

寄存器名称 : RSTC_MR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	BODIEN

15	14	13	12	11	10	9	8
–	–	–	–	ERSTL			

7	6	5	4	3	2	1	0
–	–		URSTIEN	–	–	–	URSTEN

- **URSTEN: 用户复位使能**

0 = 检测到 NRST 引脚为低电平不产生用户复位。

1 = 检测到 NRST 引脚为低电平产生用户复位。

- **URSTIEN: 用户复位中断使能**

0 = 寄存器 RSTC_SR 的位 USRTS 为 1 时对 rstc_irq 没有作用。

1 = 若 URSTEN = 0 则寄存器 RSTC_SR 的位 USRTS 为 1 时触发 rstc_irq。

- **BODIEN: 掉电检测中断使能**

0 = 寄存器 RSTC_SR 的位 BODSTS 为 1 时对 rstc_irq 没有作用。

1 = 寄存器 RSTC_SR 的位 BODSTS 为 1 时触发 rstc_irq。

- **ERSTL: 外部复位持续时间**

这个域定义了外部复位的持续时间。外部复位一直持续 $2^{(ERSTL+1)}$ 个慢速时钟周期。时间范围为 60 µs 到 2 秒。

- **KEY: 预设值**

0xA5。写入其它任何数值都将使本次写操作失败。

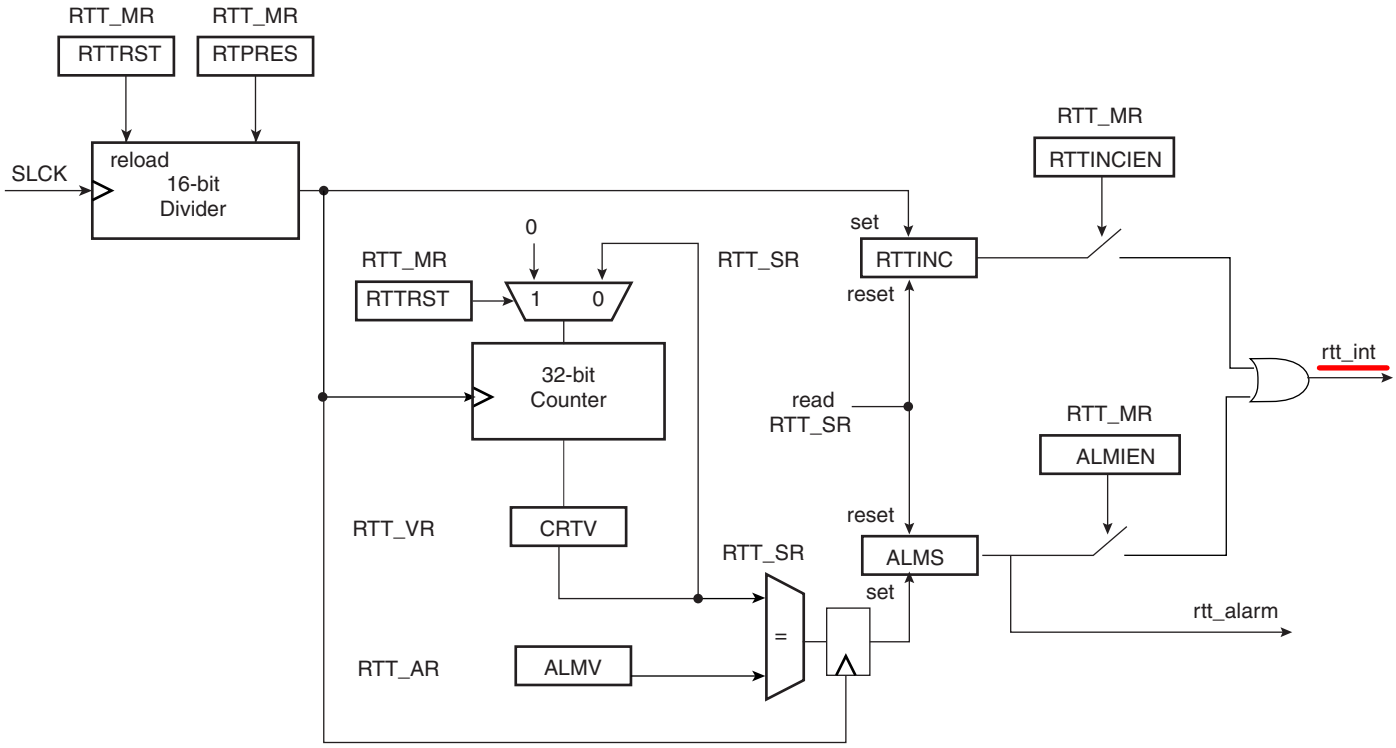
实时定时器 (RTT)

概述

实时定时器基于一个 32 位的计数器，用来记录经过的秒数。它可以产生周期性的中断或 / 和基于一个预选编好程的数值触发闹铃。

方框图

Figure 22. 实时定时器



功能描述

实时定时器用来记录经历的秒数。它基于 32 位的计数器。而这个计数器的时钟来源为慢速时钟，并经过 16 位数值的预分频。这个数值要写入实时模式寄存器 RTT_MR 的 RTPRES 域。

将 RTPRES 设置为 0x00008000 相当于给实时计数器提供 1Hz 的时钟信号 (如果慢速时钟为 32.768 Hz)。 32 位的计数器可以计 2^{32} 秒，相当于 136 年，然后计数器数值恢复为 0。

实时定时器也可以用做自由运行的基于慢速时钟的定时器。获得最佳精度的方法是将 RTPRES 设置为 1。此时提供给实时计数器的时钟信号的周期为 30.52 µs (慢速时钟为 32.768 Hz 时)。实时定时器可以应付 131072 秒，相当于 36 天。

可以在任何时候通过读取寄存器 RTT_VR 来获得实时定时器的数值 (CRTV)。因为这个数值由主时钟进行异步更新，所以建议在读取的时候连续读两次，以解决数值归零可能带来的问题。

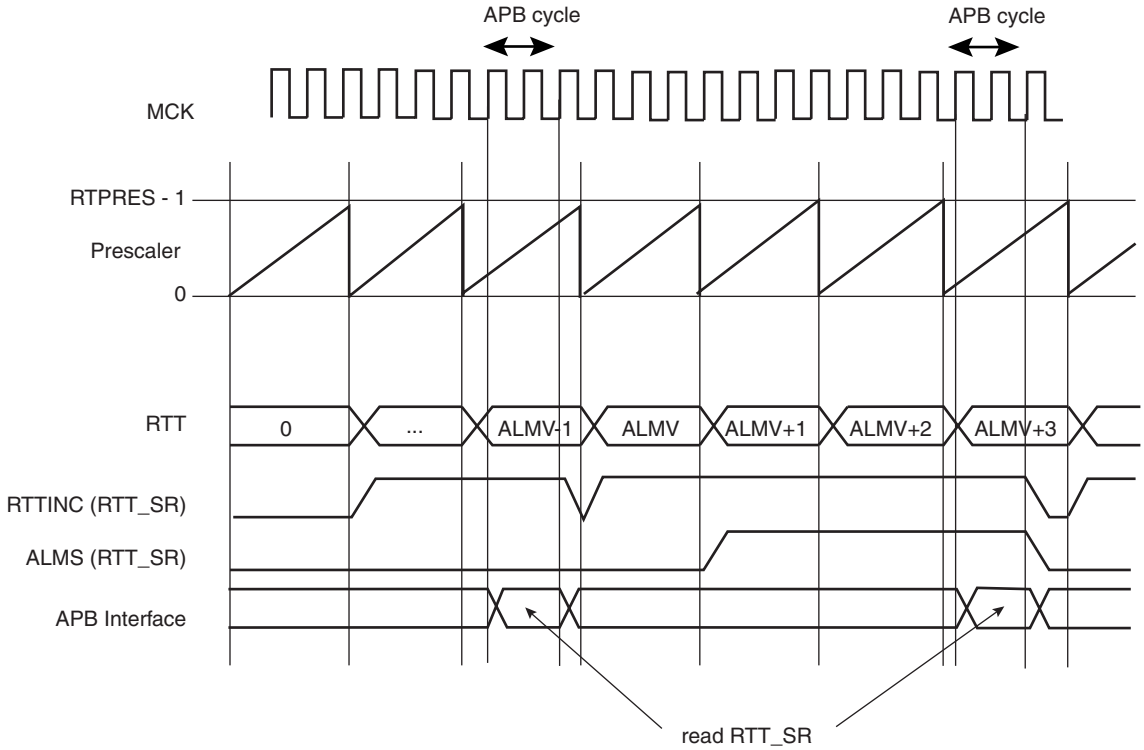
计数器的当前值与写入闹铃寄存器 RTT_AR (实时闹铃寄存器) 的数值进行比较。两个数值相等将置位寄存器 RTT_SR 的 ALMS。复位时闹铃寄存器的数值设置为最大值 0xFFFF_FFFF。

每一次实时定时器计数器数值增加时 RTT_SR 寄存器的位 RTTINC 即置位。这一位可以用来产生周期性中断。当 RTPRES 编程为 0x8000，且慢速时钟等于 32.768 Hz 时周期为一秒。

读取状态寄存器 RTT_SR 将复位 RTTINC 和 ALMS。

执行对寄存器 RTT_MR 的位 RTTRST 的写操作将立即使 RTT 重新加载及启动时钟分频器。这个操作同时也复位 32 位的计数器。

Figure 23. RTT 计数



实时定时器 (RTT) 用户接口

Table 12. 实时定时器寄存器映射

地址偏移	寄存器	名称	访问类型	复位值
0x00	模式寄存器	RTT_MR	读 / 写	0x0000_8000
0x04	闹铃寄存器	RTT_AR	读 / 写	0xFFFF_FFFF
0x08	数值寄存器	RTT_VR	只读	0x0000_0000
0x0C	状态寄存器	RTT_SR	只读	0x0000_0000

实时定时器模式寄存器

寄存器名称： RTT_MR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	RTTRST	RTTINCIEN	ALMIEN

15	14	13	12	11	10	9	8
RTPRES							

7	6	5	4	3	2	1	0
RTPRES							

- **RTPRES: 实时定时器预分频数值**

定义使实时定时器加一的 SLCK 个数。

RTPRES = 0：预分频值为 2^{16} 。

RTPRES ≠ 0：预分频值为 RTPRES。

- **ALMIEN: 中断使能**

0 = RTT_SR 寄存器的 ALMS 对中断不起作用。

1 = ALMS 将引发中断。

- **RTTINCIEN: 实时定时器增一中断使能**

0 = RTT_SR 寄存器的 RTTINC 对中断不起作用。

1 = RTTINC 将引发中断。

- **RTTRST: 实时定时器重新启动**

1 = 重新加载及重新启动时钟分频器。此操作同时复位 32 位的计数器。

实时定时器闹铃寄存器

寄存器名称： RTT_AR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
ALMV							

23	22	21	20	19	18	17	16
ALMV							

15	14	13	12	11	10	9	8
ALMV							

7	6	5	4	3	2	1	0
ALMV							

- **ALMV: 闹铃值**

定义了与实时定时器进行比较的数值，(ALMV+1)。

实时定时器数值寄存器

寄存器名称： RTT_VR

访问类型： 只读

31	30	29	28	27	26	25	24
CRTV							

23	22	21	20	19	18	17	16
CRTV							

15	14	13	12	11	10	9	8
CRTV							

7	6	5	4	3	2	1	0
CRTV							

- **CRTV: 当前实时数值**

返回实时定时器的当前数值。

实时定时器状态寄存器

寄存器名称： RTT_SR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	–	–	RTTINC	ALMS

- **ALMS: 实时状态**

0 = 自上一次读取寄存器 RTT_SR 之后实时没有发生过。

1 = 自上一次读取寄存器 RTT_SR 之后发生了实时。

- **RTTINC: 实时定时器增加**

0 = 自上一次读取寄存器 RTT_SR 之后实时定时器没有增加。

1 = 自上一次读取寄存器 RTT_SR 之后实时定时器增加了。

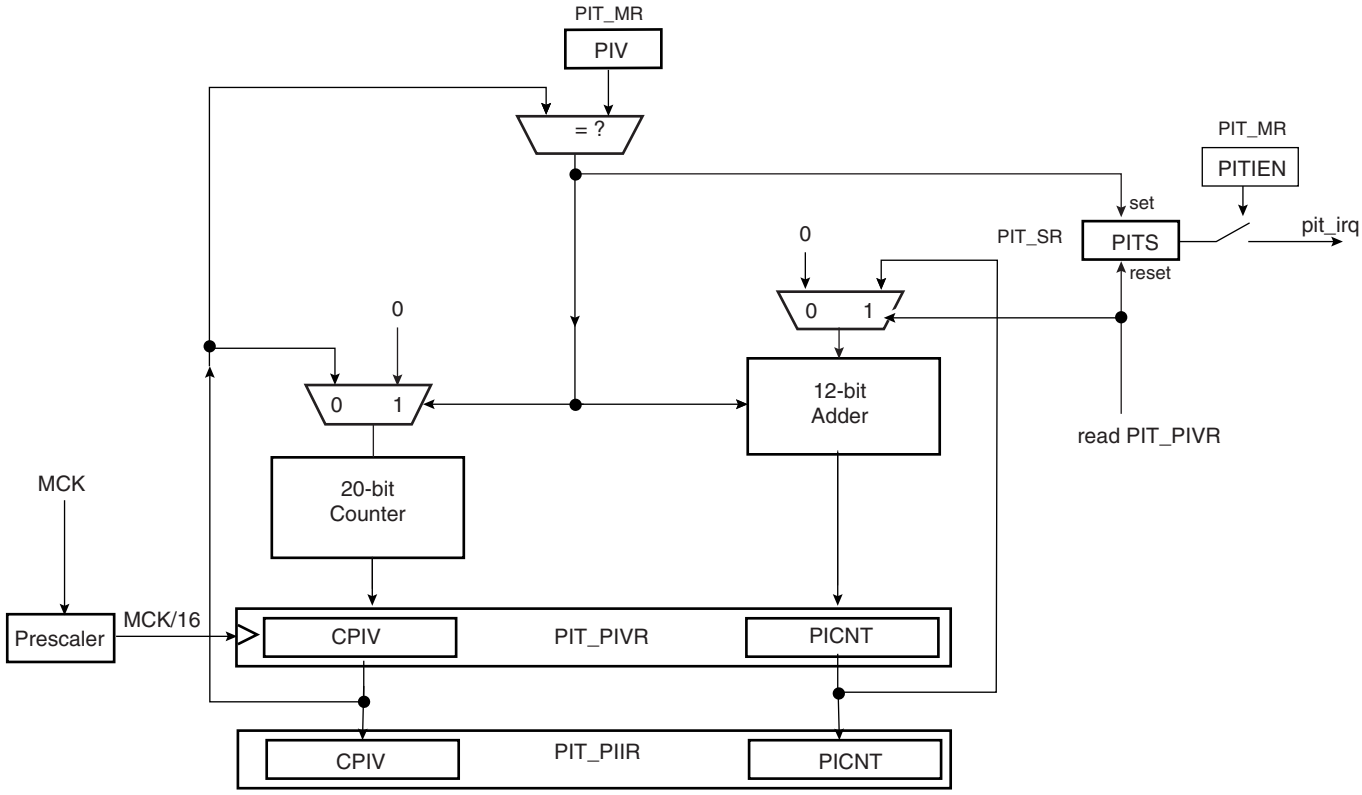
周期性间隔定时器 (PIT)

概述

周期性间隔定时器 (PIT) 为操作系统的调度程序提供时间间隔中断。这个定时器设计的目的是提供最大的精确度和高效率的管理，即使对要求长响应时间的系统也是如此。

方框图

Figure 24. 周期性间隔定时器



功能说明

周期性间隔定时器的目的是为操作系统提供周期性的中断。

PIT 有一个可编程的溢出计数器和读后即复位的特性。它基于两个计数器：一个 20 位的 CPIV 计数器和一个 12 位的 PICNT 计数器。两个计数器的时钟都是主时钟的 1/16。

20位的CPIV计数器从0开始计数，直到模式寄存器 PIT_MR的PIV域定义的溢出数值为止。CPIV 计到这个数值后即复位为 0，同时周期性间隔计数器 PICNT 加一。然后状态寄存器 PIT_SR 的位 PITS 置位并触发中断，只要此时中断是使能的 (PIT_MR 的位 PITIEN)。

将新的数值赋予给寄存器 PIT_MR 的 PIV 域不会复位 / 重启动计数器。

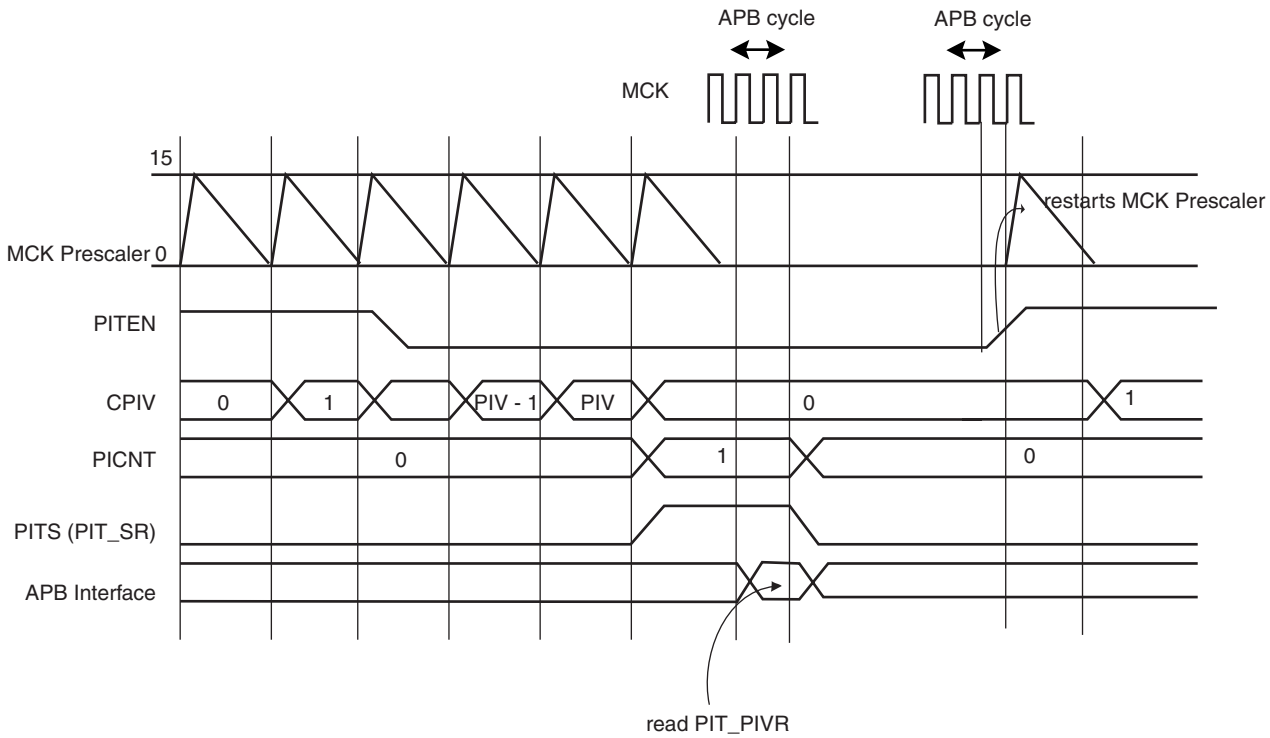
通过读取周期性间隔数值寄存器PIT_PIVR获得CPIV和PICNT的数值之后，溢出计数器(PICNT)复位， PITS 清零，从而确认中断。 PICNT 的数值表示自上一次读取 PIT_PIVR 之后发生的周期性间隔次数。

通过读取周期性间隔映像寄存器 PIT_PIIR 获得 CPIV 和 PICNT 的数值对计数器的 CPIV 和 PICNT，以及 PITS 没有影响。例如，软件剖析工具可以读取 PIT_PIIR 而不会清除任何已经发生但还未处理的中断，而定时器中断通过读 PIT_PIVR 清除中断。

通过寄存器 PIT_MR 的 PITEN 位可以使能 / 禁止 PIT(复位时处于禁止状态)。只有在 CPIV 为 0 时操作 PITEN 才有作用。Figure 25 演示了 PIT 的计数方式。PIT 使能位复位后 (PITEN= 0)，CPIV 继续计数，直到达到 PIV 的数值，然后再复位。 PITEN 置位后 PIT 才能重新开始计数。

在内核进入调试状态时 PIT 停止。

Figure 25. 利用 PITEN 使能 / 禁用 PIT



周期性间隔定时器 (PIT) 用户接口

Table 13. 周期性间隔定时器 (PIT) 寄存器映射

地址偏移	寄存器	名称	访问类型	复位值
0x00	模式寄存器	PIT_MR	读 / 写	0x000F_FFFF
0x04	状态寄存器	PIT_SR	只读	0x0000_0000
0x08	周期性间隔数值寄存器	PIT_PIVR	只读	0x0000_0000
0x0C	周期性间隔映像寄存器	PIT_PIIR	只读	0x0000_0000

周期性间隔定时器模式寄存器

寄存器名称 : PIT_MR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	PITIEN	PITEN

23	22	21	20	19	18	17	16
–	–	–	–	PIV			

15	14	13	12	11	10	9	8
PIV							

7	6	5	4	3	2	1	0
PIV							

- **PIV: 周期性间隔数值**

定义了与周期性间隔寄存器 20 位计数器 CPIV 比较的数值。周期等于 PIV + 1。

- **PITEN: 使能周期性间隔定时器**

0 = 计数值达到 PIV 后周期性间隔寄存器被禁用。

1 = 周期性间隔寄存器使能。

- **PITIEN: 使能周期性间隔定时器中断**

0 = PIT_SR 寄存器的 PITS 对中断没有影响。

1 = PIT_SR 寄存器的 PITS 可以触发中断。

周期性间隔定时器状态寄存器

寄存器名称： PIT_SR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	PITS

- **PITS: 周期性间隔定时器状态**

0 = 自上次读取 PIT_PIVR 之后周期性间隔定时器还没有计数到 PIV。

1 = 自上次读取 PIT_PIVR 之后周期性间隔定时器已经计数到 PIV。

周期性间隔定时器数值寄存器

寄存器名称 : PIT_PIVR

访问类型 : 只读

31	30	29	28	27	26	25	24
PICNT							

23	22	21	20	19	18	17	16
PICNT				CPIV			

15	14	13	12	11	10	9	8
CPIV							

7	6	5	4	3	2	1	0
CPIV							

读取该寄存器将清除寄存器 PIT_SR 的 PITS。

- **CPIV: 当前周期性间隔数值**

返回周期性间隔定时器的当前数值。

- **PICNT: 周期性间隔计数器**

返回自上一次读取 PIT_PIVR 之后发生的周期性间隔次数。

周期性间隔定时器映像寄存器

寄存器名称 : PIT_PIIR

访问类型 : 只读

31	30	29	28	27	26	25	24
PICNT							

23	22	21	20	19	18	17	16
PICNT				CPIV			

15	14	13	12	11	10	9	8
CPIV							

7	6	5	4	3	2	1	0
CPIV							

- **CPIV: 当前周期性间隔数值**

返回周期性间隔定时器的当前数值。

- **PICNT: 周期性间隔计数器**

返回自上一次读取 PIT_PIVR 之后发生的周期性间隔次数。

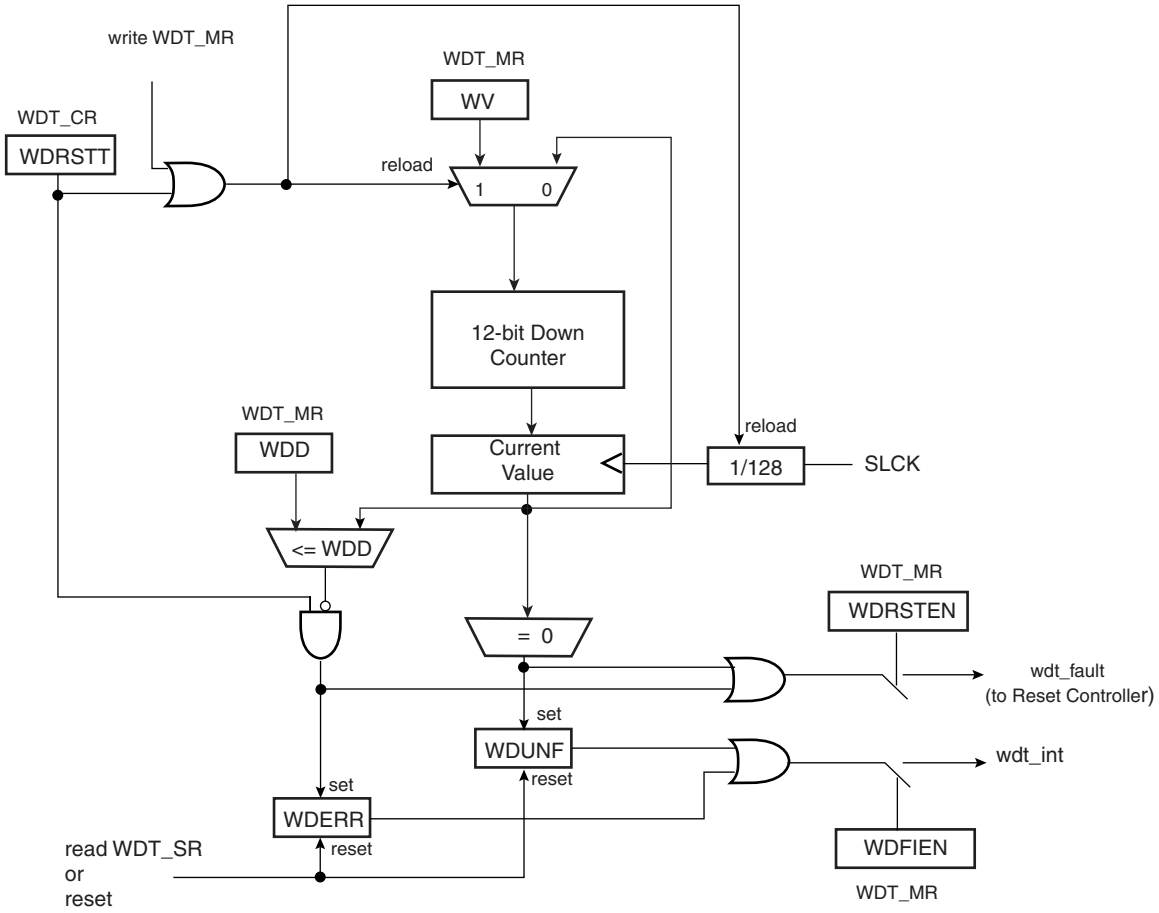
看门狗定时器 (WDT)

概述

看门狗定时器可以用来防止由于软件陷于死循环而导致的系统死锁。它具有一个 12 位的向下计数器，使得看门狗周期可以达到 16 秒 (慢速时钟， 32.768 kHz)。它可以产生通常的复位，或者仅仅是处理器复位。此外，当处理器处于调试模式或空闲模式时看门狗可以被禁止。

方框图

Figure 26. 看门狗定时器框图



功能描述

看门狗定时器可以用来防止由于软件陷于死循环而导致的系统死锁。它的电源是 VDDCORE。处理器复位后它从初始值重新开始启动。

看门狗基于一个 12 位的向下计数器，加载的数据可以通过模式寄存器 WDT_MR 的 WV 域来定义。如果使用经过 128 分频的慢速时钟来驱动看门狗定时器，溢出周期达到最大值 16 秒 (慢速时钟的典型值为 32.768 kHz)。

处理器复位之后， WV 的数值为 0xFFF，对应于计数器的最大值，并使能了外部复位 (Backup 复位时 WDRSTEN 为 1)。也就是说，复位之后看门狗就运行了。如果用户程序没有使用看门狗，则必须禁止它 (置位 WDT_MR 寄存器的位 WDDIS)，否则必须对它进行编程以满足应用要求。

看门狗模式寄存器 (WDT_MR) 只能写一次。只有处理器复位才可以复位它。对 WDT_MR 执行写操作可以把最后编程的模式参数加载到定时器。

在普通的操作中，用户需要通过置位控制寄存器 WDT_CR 的位 WDRSTT 来定时地重新加载看门狗，以防止定时器溢出。计数器将立即由 WDT_MR 重新加载并重新启动，慢速时钟的 128 分频器也被复位及重新启动。 WDT_CR 是写保护的。因此，如果预设值不正确，对 WDT_CR 的写操作是没有作用的。如果发生了计数器溢出，并且模式寄存器 WDT_MR 的 WDRSTEN 为 1，则连接到复位控制器的“wdt_fault”信号生效，看门狗状态寄存器WDT_SR的位WDUNF也被设置为 1。

为了防止软件死锁时持续不断地触发看门狗，看门狗的重新加载必须在由 0和WDT_MR的WDD 定义的时间窗之内发生：

0 ≤WDT ≤WDD：执行对 WDRSTT 的写操作将使看门狗定时器重新启动。

任何在 [WDV : WDD] 之间重新启动看门狗定时器的企图都将导致看门狗错误，即使此时看门狗时禁止的：WDT_SR 的位 WDERR 置位，连接到复位控制器的 “wdt_fault” 信号生效。

要注意的是，当 WDD 等于或大于 WDV 时，这个特性即被禁止。这个配置允许看门狗定时器在 [0 : WDV] 的整个区间都可以重新启动而不产生错误。这也是芯片复位时的缺省配置 (WDD 与 WDV 的数值时相等的)。

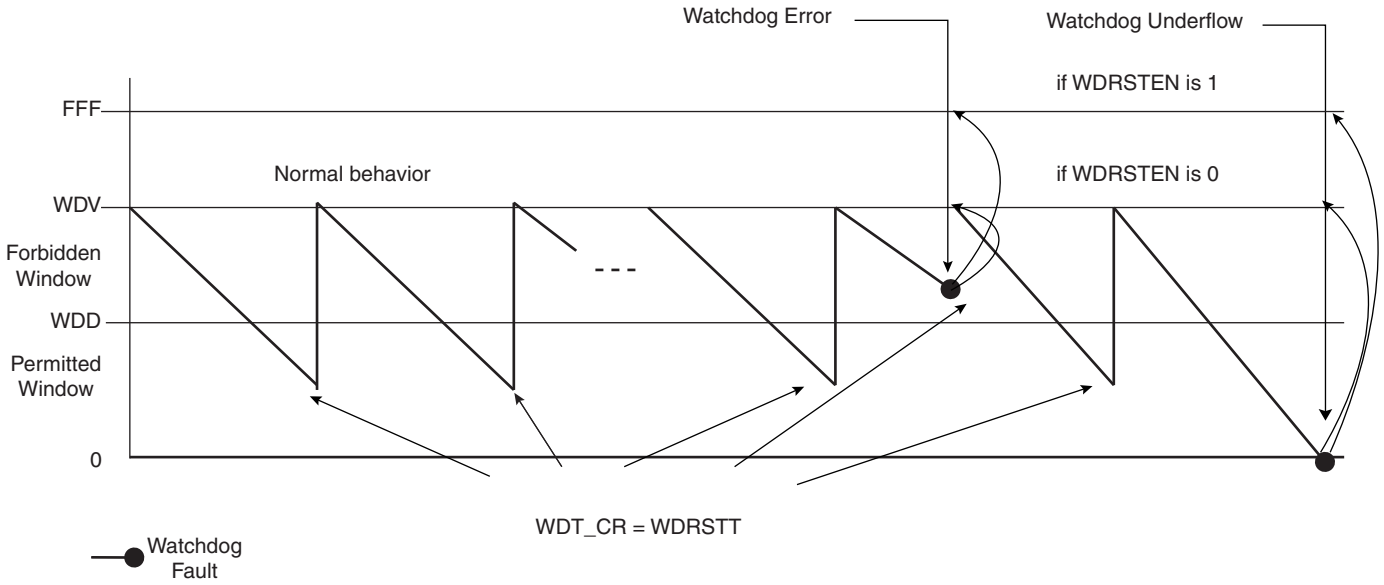
若模式寄存器的 WDFIEN 为 1，状态位 WDUNF (看门狗溢出) 和 WDERR (看门狗错误) 置位将触发中断。如果 WDRSTEN 同时也为 1，则连接到复位控制器的 “wdt_fault” 信号将引起看门狗复位。在这种情况下，处理器和看门狗定时器复位， WDERR 及 WDUNF 标志被清零。

如果复位已经产生，或是读访问了寄存器 WDT_SR，则状态位被清零，中断被清除，输送到复位控制器的 “wdt_fault” 信号不再有效。

执行对 WDT_MR 的写操作将重新加载向下计数器，并使其重新启动。

当处理器处于调试状态或空闲模式时，根据寄存器 WDT_MR 的 WDIDLEHLT 和 WDDBGHLT 的设置，计数器可以被停止。

Figure 27. 看门狗的工作行为



看门狗定时器 (WDT) 用户接口

Table 14. 看门狗定时器 (WDT) 寄存器映射

地址偏移	寄存器	名称	访问类型	复位值
0x00	控制寄存器	WDT_CR	只写	-
0x04	模式寄存器	WDT_MR	读 / 写，一次	0x3FFF_2FFF
0x08	状态寄存器	WDT_SR	只读	0x0000_0000

看门狗定时器控制寄存器

寄存器名称： WDT_CR

访问类型： 只写

31	30	29	28	27	26	25	24
KEY							
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	WDRSTT

- **WDRSTT: 看门狗重新启动**

0：无效。

1：重新启动看门狗。

- **KEY: 预设值**

0xA5。写入其它任何数值都将使本次写操作失败。

看门狗定时器模式寄存器

寄存器名称： WDT_MR

访问类型： 读 / 写，一次

31	30	29	28	27	26	25	24
		WDIDLEHLT	WDDBGHLT	WDD			
23	22	21	20	19	18	17	16
WDD							
15	14	13	12	11	10	9	8
WDDIS	WDRPROC	WDRSTEN	WDFIEN	WDV			
7	6	5	4	3	2	1	0
WDV							

- **WDV: 看门狗计数器的数值**

定义了加载到 12 位看门狗计数器的数值。

- **WDFIEN: 使能看门狗故障中断**

0：看门狗故障 (溢出或错误) 对中断不产生影响。

1：看门狗故障 (溢出或错误) 将产生中断。

- **WDRSTEN: 使能看门狗复位**

0：看门狗故障 (溢出或错误) 对复位没有影响。

1：看门狗故障 (溢出或错误) 将引发看门狗复位。

- **WDRPROC: 看门狗复位处理器**

0：若 WDRSTEN 为 1，则看门狗故障 (溢出或错误) 将引发所有的复位。

1：若 WDRSTEN 为 1，则看门狗故障 (溢出或错误) 只引发处理器复位。

- **WDD: 看门狗 Delta 数值**

定义了重新加载看门狗定时器所允许的范围。

如果看门狗定时器的数值小于等于 WDD，置位 WDT_CR 的 WDRSTT 将使定时器重新启动。

如果看门狗定时器的数值大于 WDD，置位 WDT_CR 的 WDRSTT 将产生看门狗错误。

- **WDDBGHLT: 看门狗调试停止**

0：处理器处于调试状态时看门狗仍然运行。

1：处理器处于调试状态时看门狗停止。

- **WDIDLEHLT: 看门狗空闲停止**

0：系统处于空闲状态时看门狗仍然运行。

1：系统处于空闲状态时看门狗停止。

- **WDDIS: 禁止看门狗**

0：使能看门狗定时器。

1：禁止看门狗定时器。

看门狗定时器状态寄存器

寄存器名称： WDT_SR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	WDERR	WDUNF

- **WDUNF: 看门狗溢出**

0：自上一次读取 WDT_SR 之后没有发生看门狗溢出。

1：自上一次读取 WDT_SR 之后至少发生了一次看门狗溢出。

- **WDERR: 看门狗错误**

0：自上一次读取 WDT_SR 之后没有发生看门狗错误。

1：自上一次读取 WDT_SR 之后至少发生了一次看门狗错误。

电压调节器模式控制器 (VREG)

概述

电压调节器模式控制器包括一个读 / 写寄存器 - 电压调节器模式寄存器。它相对系统控制器的偏移地址为 0x60。

这个寄存器控制着电压调节器的模式。置位 PSTDBY 将使电压调节器进入待机模式或低功耗模式。一旦复位， PSTDBY 即清零，从而使电压调节器进入正常模式。

电压调节器电源控制器 (VREG) 用户接口

Table 15. 电压调节器电源控制器寄存器映射

地址偏移	寄存器	名称	访问类型	复位值
0x60	电压调节器模式寄存器	VREG_MR	读 / 写	0x0

电压调节器模式寄存器

寄存器名称 : VREG_MR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	PSTDBY

- **PSTDBY: 周期性间隔数值**

0 = 普通模式。

1 = 待机模式 (低功耗模式)。

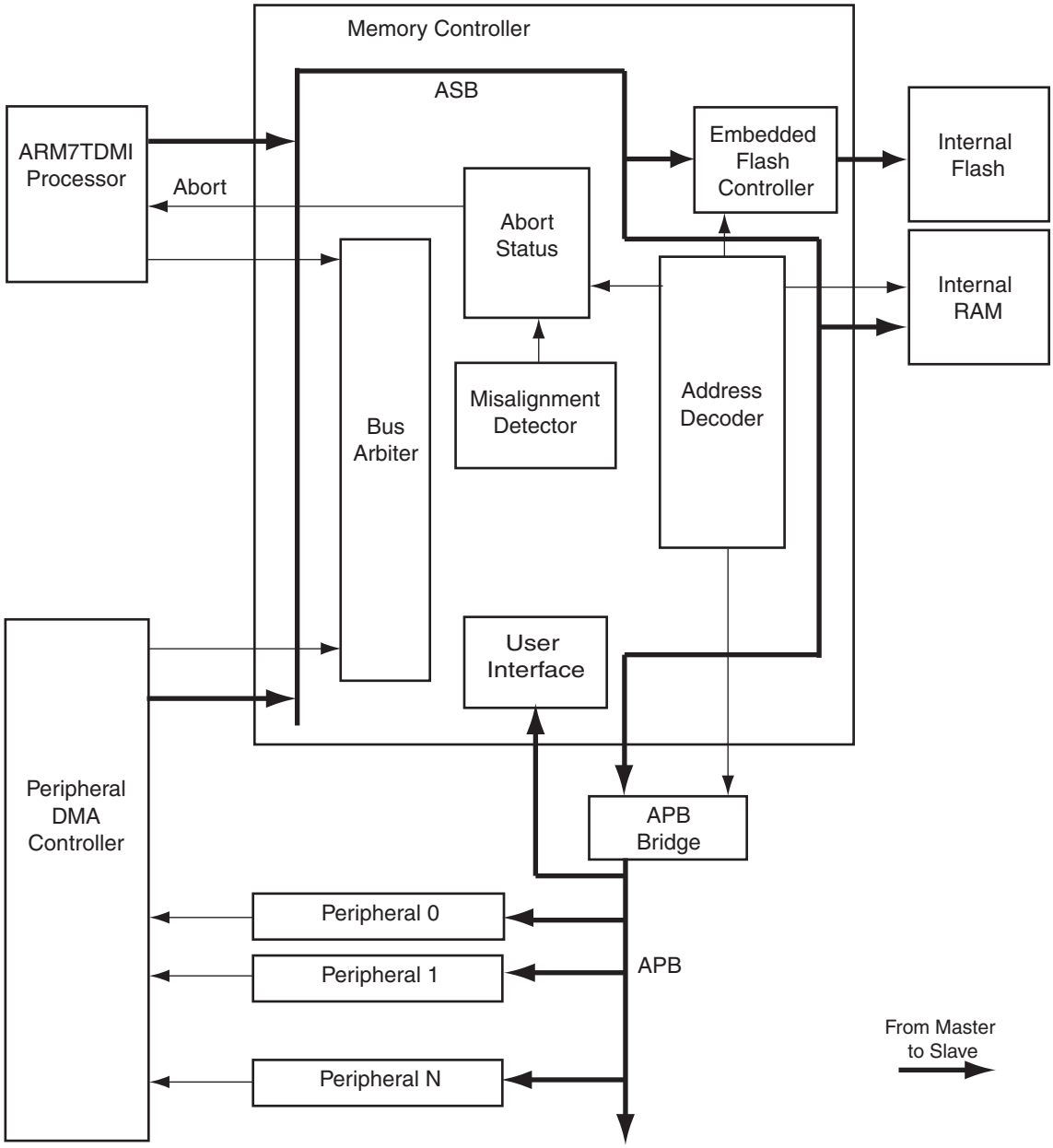
存储器控制器 (MC)

概述

存储器控制器 (MC) 管理 ASB 总线，并控制主机的访问请求。这里主机指的使 ARM7TDMI 处理器以及外设数据控制器。它具有一个简单的总线仲裁器，地址译码器，异常中断状态，地址未对齐检测器和嵌入式 Flash 控制器。

方框图

Figure 28. 存储器控制器框图



功能说明

存储器控制器处理 ASB 总线，并对主机的访问进行仲裁。

其组成部分为：

- 总线仲裁器
- 地址译码器
- 异常中断状态
- 地址未对齐检测器
- 嵌入式 Flash 控制器

MC 只能够处理 little-endian 模式的访问。主机也只能够工作于 little-endian 模式。

总线仲裁器

存储器控制器有一个简单的、硬连接优先级总线仲裁器，控制两个主机对总线的访问。外设数据控制器的优先级比 ARM 处理器的要高。

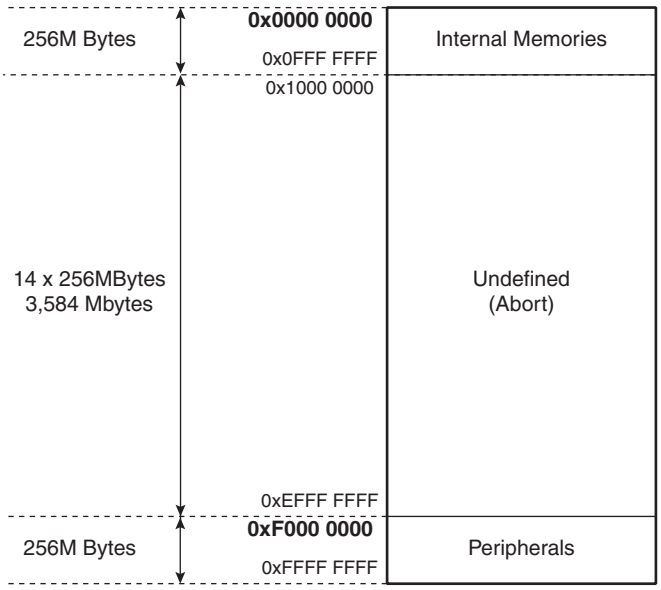
地址译码器

存储器控制器的地址译码器首先对 32 位地址总线的高 4 位进行解码，并定义了 3 个不同的地址范围：

- 256M 字节的内部存储器
- 为片内外设所保留的 256M 字节地址空间
- 未定义的 3584M 字节表现为 14 个 256M 字节的存储区。访问这个区域将导致异常中断

Figure 29 给出了 256M 字节存储区的分配情况。

Figure 29. 存储区



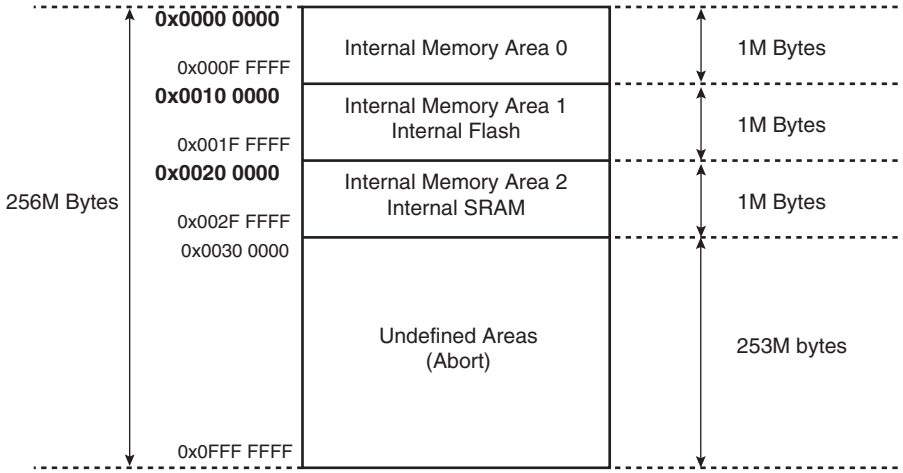
片内存储器映射

在内部存储器地址空间里，地址译码器将增加 8 个比特的译码，以便为嵌入式存储器分配 1M 字节的地址空间。

分配的存储器位于 1M 字节的地址空间，在这个访问空间里可能会重复 n 次， n 等于 1M 字节除以存储器大小。

访问地址没有定义时，地址译码器将返回异常中断到主机。

Figure 30. 内部存储器映射



内部存储器区域 0

内部存储器区域 0 的前 32 个字节包含了 ARM 处理器的例外向量，尤其是位于地址 0x0 的复位向量。

执行重映射 (remap) 命令之前，片内 Flash 映射于内部存储器区域 0，因此 ARM7TDMI 可以访问到位于 Flash 的可执行指令。重映射之后，原来位于地址 0x0020 0000 的片内 SRAM 映射到内部存储器区域 0。它同时还可以原来的地址进行访问。

重映射命令

重映射之后，片内 SRAM 可以通过 内部存储器区域 0 进行访问。

由于 ARM 向量 (复位，异常，数据异常，预取异常，未定义指令，中断和快速中断) 从地址 0x0 映射到地址 0x20，重映射命令允许用户通过软件动态地对这些向量重新进行定义。

重映射命令通过存储器控制器用户接口进行控制，方法是将寄存器 MC_RCR (重映射控制寄存器) 的 RCB 置位。

再一次置位 MC_RCR 的 RCB 即取消重映射命令。这个功能可以将芯片置于与复位之后相同的配置，从而简化用户自定义启动序列的调试。

异常状态

异常发生的原因有三个：

- 访问未定义的地址
- 访问未对齐的地址

异常发生时，所有的主机都将接收到一个信号，不论是哪一个主机产生了这个访问。但是只有 ARM7TDMI 会处理此信号，而且是在这个访问由它自己产生的条件之下。外设数据控制器不处理异常输入信号。 Figure 28 没有给出信号的连接关系。

为了便利操作系统的调试或错误分析，存储器控制器集成了异常状态寄存器集。

完整的 32 位异常地址保存于 MC_AASR，访问参数则保存于 MC_ASR，其内容有：

- 请求的变量位宽 (ABTSZ)
- 访问类型，看其是数据读 / 写访问还是代码访问 (ABTTYP)
- 访问的是未定义的地址 (UNDADD) 还是未对齐的地址 (MISADD)
- 导致异常的访问来源 (MST0 和 MST1)
- 自从上一次读取这个寄存器后是否每一个主机都发生了异常 (SVMST0 和 SVMST1)，除非这个信息已经加载到 MSTx

发生来自处理器的数据异常时，数据地址得以保存。这非常有用，因为搜索哪个地址引起异常要求进行反汇编并明了处理器的上下文。

发生预取异常时，由于预取发生在流水线中，地址可能已经变了。只有在执行读指令时 ARM 处理器才会理会预取异常，而此时可能已经发生几次异常了。因此最好使用 ARM 处理器的异常链接寄存器。

嵌入式 Flash 控制器

嵌入式 Flash 控制器被加入存储器控制器以保证 Flash 模块与 32 位内部总线的接口。通过其 32 位的缓冲器可以提高 Thumb 模式下代码获取的性能。它同时还通过完整的命令集管理着 Flash 的编程、擦除、加锁和解锁。

不对齐检测器

存储器控制器的未对齐检测器可以检查内存访问的一致性。

不论内存访问是由哪一个主机发起，检测器将检查每一次内存访问的总线宽度以及位 0 和位 1。如果以字 (32 位) 来访问，而且位 0 和位 1 不为 0，或者以半自字 (16 位) 来访问，而位 0 不为 0，则异常返回到主机，此次访问被取消。要注意的是，当 ARM 处理器在获取指令时不执行这种检查。

地址不对齐通常是由软件缺陷引起的，会引发错误的指针操作。这些软件缺陷在调试阶段特别难以检测。

由于请求的地址保存于异常状态寄存器，而引起地址不对齐的指令保存于异常链接寄存器，检测及修正这类软件缺陷因此得到了简化。

存储器控制器 (MC) 用户接口

基地址：0xFFFFFF00

Table 16. 存储器控制器 (MC) 寄存器映射

偏移地址	寄存器	名称	访问类型	复位状态
0x00	MC 重映射控制寄存器	MC_RCR	只写	
0x04	MC 异常状态寄存器	MC_ASR	只读	0x0
0x08	MC 异常地址状态寄存器	MC_AASR	只读	0x0
0x0C-0x5C	保留	–	–	–
0x60	EFC 配置寄存器	参见 “ 嵌入式 Flash 控制器 (EFC)” on page 93.		

MC 重映射控制寄存器

寄存器名称 :MC_RCR

访问类型： 只写

偏移地址： 0x00

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	RCB

- **RCB: 重映射命令位**

0：无效。

1：这一位的作用象是一个切换开关：写 1 将交替地执行 / 取消第零页储存器的重映射。

MC 异常状态寄存器

寄存器名称： MC_ASR

访问类型：只读

复位值： 0x0

偏移地址： 0x04

31	30	29	28	27	26	25	24
–	–	–	–	–	–	SVMST1	SVMST0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	MST1	MST0
15	14	13	12	11	10	9	8
–	–	–	–	ABTTYP		ABTSZ	
7	6	5	4	3	2	1	0
–	–	–	–	–	–	MISADD	UNDADD

- **UNDADD: 未定义的地址异常状态**

0: 上一个异常不是由于执行了对未定义地址的访问而引起的。

1: 上一个异常是由于执行了对未定义地址的访问而引起的。

- **MISADD: 未对齐的地址异常状态**

0: 上一个异常不是由于地址未对齐引起的。

1: 上一个异常是由于地址未对齐引起的。

- **ABTSZ: 异常变量位宽状态**

ABTSZ		异常变量位宽
0	0	字节
0	1	半字
1	0	字
1	1	保留

- **ABTTYP: 异常类型状态**

ABTTYP		异常类型
0	0	读数据
0	1	写数据
1	0	获取代码
1	1	保留

- **MST0: ARM7TDMI 异常来源**

0：上一个异常不是由于 ARM7TDMI 引起的。

1：上一个异常是由于 ARM7TDMI 引起的。

- **MST1: PDC 异常来源**

0：上一个异常不是由于 PDC 引起的。

1：上一个异常是由于 PDC 引起的。

- **SVMST0: 保留的 ARM7TDMI 异常来源**

0：自上一次读取 MC_ASR，或是通过 MST0 得到通知之后，没有发生由于 ARM7TDMI 造成的异常。

1：自上一次读取 MC_ASR 之后，至少发生了一次由于 ARM7TDMI 造成的异常。

- **SVMST1: 保留的 PDC 异常来源**

0：自上一次读取 MC_ASR，或是通过 MST1 得到通知之后，没有发生由于 PDC 造成的异常。

1：自上一次读取 MC_ASR 之后，至少发生了一次由于 PDC 造成的异常。

MC 异常地址状态寄存器

寄存器名称： MC_AASR

访问类型：只读

复位值： 0x0

偏移地址： 0x08

31	30	29	28	27	26	25	24
ABTADD							
23	22	21	20	19	18	17	16
ABTADD							
15	14	13	12	11	10	9	8
ABTADD							
7	6	5	4	3	2	1	0
ABTADD							

- **ABTADD: 异常地址**

包含了上一次异常访问的地址。

嵌入式 Flash 控制器 (EFC)

概述

嵌入式 Flash 控制器 (EFC) 是存储器控制器的一部分，确保 Flash 块与 32 位内部总线的接口。通过其 32 位缓冲器系统，EFC 提高了 Thumb 模式下代码获取的性能。它同时还通过一套完整的命令集管理 Flash 的编程、擦除、加锁和解锁。

功能说明

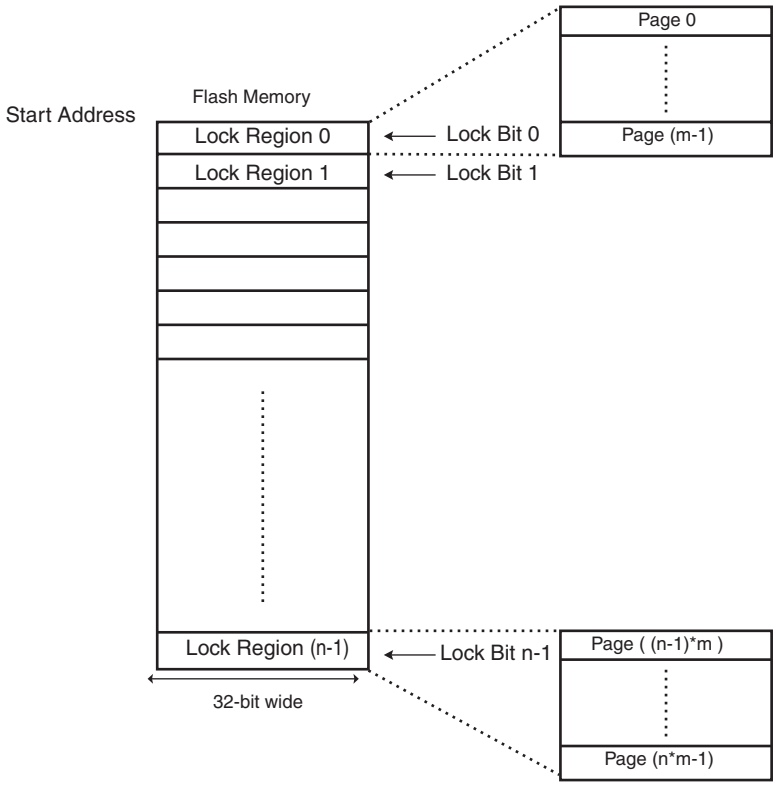
嵌入式 Flash 的组织形式

嵌入式 Flash 与 32 位内部总线直接接口。它具体由下面几个接口组成：

- 一个存储面由几个相同大小的页面构成。
- 两个 32 位读缓冲器用来优化代码访问 (See “ 读操作 ” on page 94.)。
- 一个写缓冲区用来管理页面编程。写缓冲区与页面大小相同。这个缓冲区是只写的，而且可以在 1 M 字节地址空间里进行访问。所以每个字都可以写到它的最终地址 (See “ 写操作 ” on page 96.)。
- 几个锁定位用来保护锁定区域的写和擦除操作。锁定区域由几个连续的页面组成，每个锁定区域具有自己的锁定位。
- 几个通用 NVM 位。每个位控制一个特定的功能。具体请参考产品定义一节。

嵌入式 Flash 大小，页面大小和锁定区域组织在产品定义一节有描述。

Figure 31. 嵌入式 Flash 存储器映射



读操作

一个优化的控制器管理着嵌入式 Flash 的读操作。系统增加了两个 32 位的缓冲器。其作用是访问第二个读操作的地址，从而提高了处理器运行于 Thumb(16 位指令集) 模式时的性能。图示请参见 Figure 32， Figure 33 和 Figure 34。

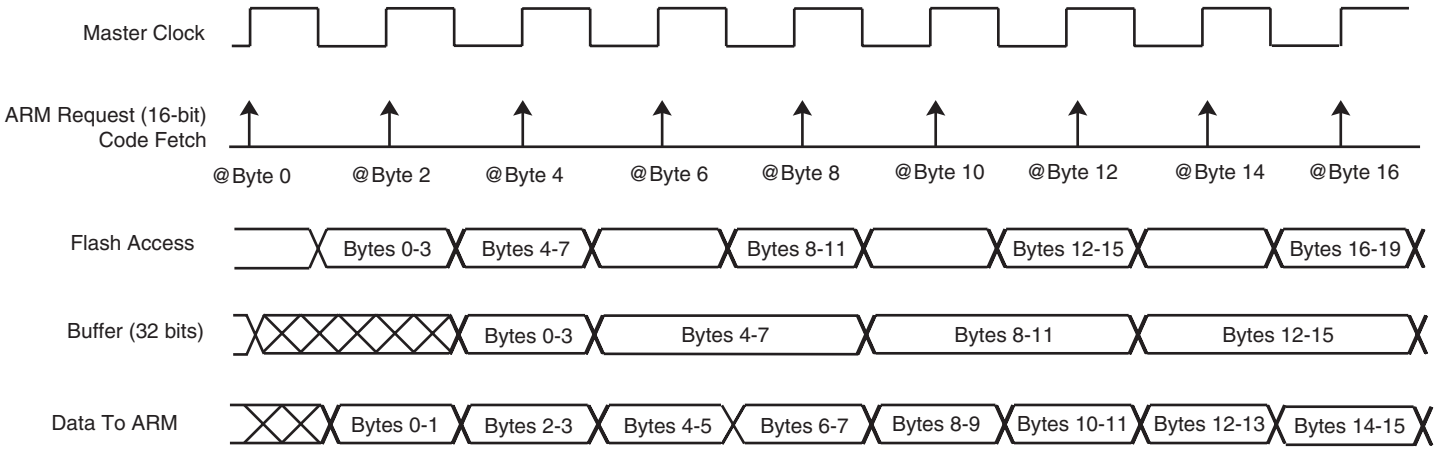
此优化只对代码获取有效，对数据无效。

读操作可以利用等待状态。通过编程 Flash 模式寄存器 MC_FMR (见 “MC Flash 模式寄存器 ” on page 102)的FWS (Flash等待状态) ,最多可以插入3个等待状态。FWS为0表示对片内Flash进行单周期访问。

Flash 存储器可以 8、 16 和 32 位的方式进行访问。

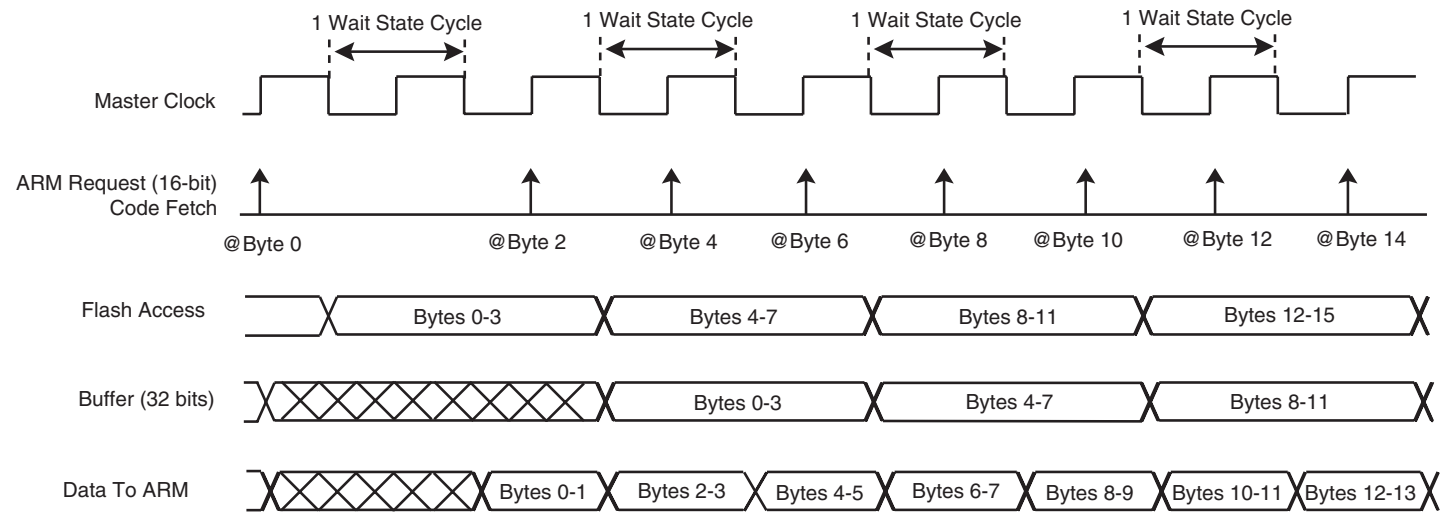
由于 Flash 模块比保留的片内存储区地址空间要小，访问这个地址空间时 Flash 将重复出现。

Figure 32. 代码读取优化， Thumb 模式， FWS = 0



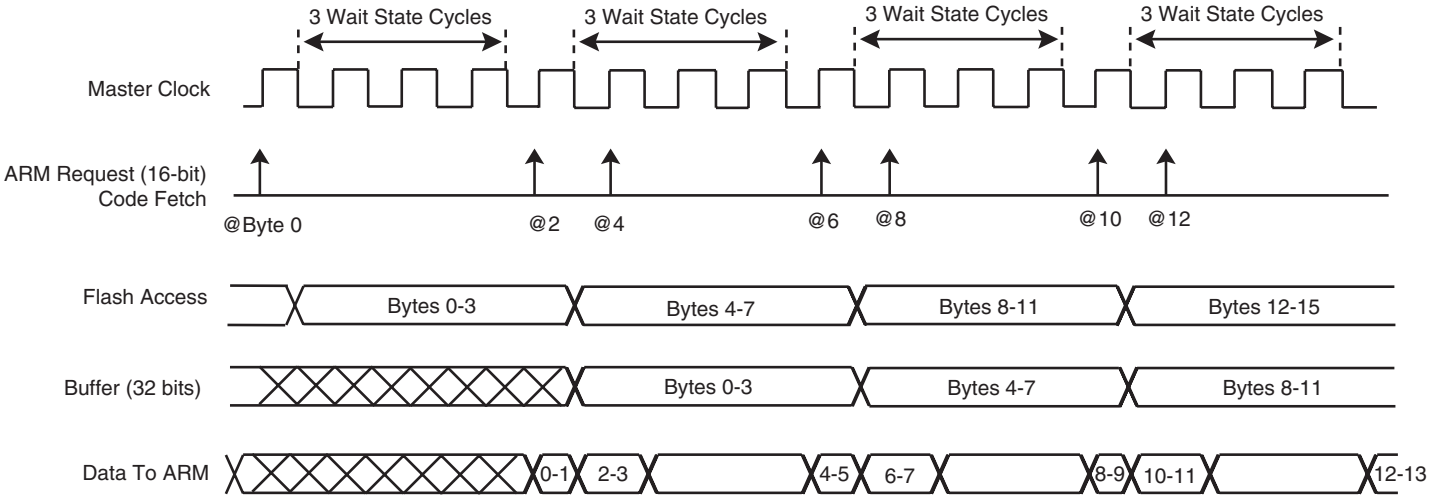
Note: FWS 为 0 时，所有的访问都是单周期访问。

Figure 33. 代码读取优化， Thumb 模式， FWS = 1



Note: FWS 为 1 且执行序列读时，所有的访问都是单周期访问 (除了第一次访问)。

Figure 34. 代码读取优化， Thumb 模式， FWS = 3



Note: FWS为2或3，且执行序列读时，第一个访问需要FWS个周期，第二个访问需要一个周期，第三个访问需要FWS个周期，第四个访问需要一个周期，依此类推。

写操作

通过只写锁存缓冲器可以实现对片内保留的 Flash 存储空间的写操作。但是写操作只利用了最低的8位地址，因此访问整个地址空间时最低地址位，在内部存储器地址空间中反转并重复1024次。

通过编程存储器保护单元可以禁止写操作。

以 8 位和 16 位方式进行的写操作是不允许的，可能导致不可预期的数据毁坏。

执行写操作时的等待周期等于读等待周期 + 1，除了 FWS = 3 (见 “MC Flash 模式寄存器 ” on page 102) 的情况。

Flash 命令

嵌入式 Flash 控制器提供了一套命令来管理 flash 存储器的编程、扇区加锁和解锁、连续编程与上锁，以及全片擦除。

Table 17. 命令集

命令	数值	助记符
页写	0x01	WP
设置锁定位	0x02	SLB
页写和锁定	0x03	WPL
清除锁定位	0x04	CLB
全片擦除	0x08	EA
设置通用 NVM 位	0x0B	SGPB
清除通用 NVM 位	0x0D	CGPB
设定安全位	0x0F	SSB

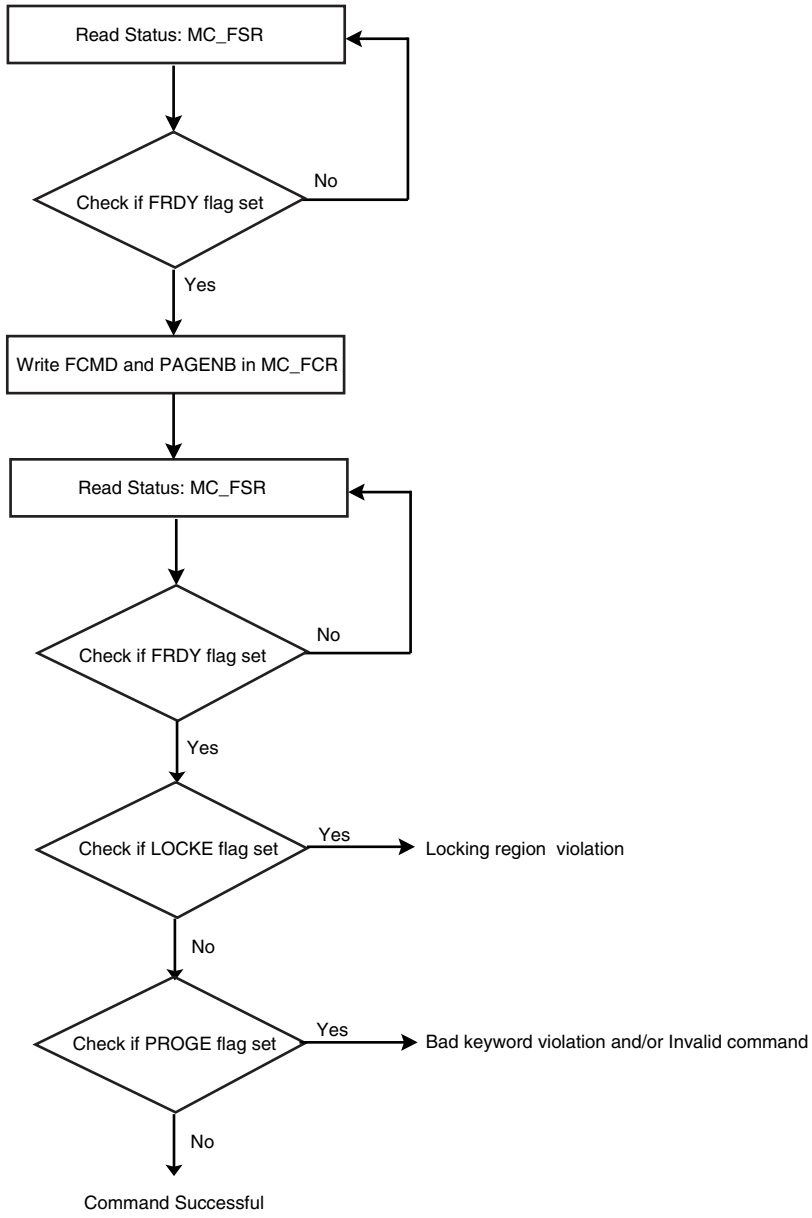
执行命令之前首先要对 MC_FCR 寄存器的 FCMD 域赋以命令编号。一旦执行了 MC_FCR 寄存器的写操作，FRDY 标志自动清零。一旦命令执行结束 FRDY 标志又自动被置位。如果此时相应的中断使能，存储器控制器的中断线即被激活。

所有的命令都通过相同的关键字进行保护。这个关键字必须写入 MC_FCR 寄存器的高 8 位。

如果关键字不正确，将数据写入 MC_FCR 对存储器没有任何影响；除了 MC_FSR 寄存器的 PROGE 标志置位。读取 MC_FSR 即可将此标志清零。

如果当前命令试图写入或擦除保护区域的某一页，此命令不会对存储器产生任何影响，除了 MC_FSR 寄存器的 FLOCKE 标志置位。读取 MC_FSR 即可将此标志清零。

Figure 35. 命令状态图



为了保证对 Flash 的正确操作，Flash 模式寄存器 MC_FMR 的 Flash 微秒周期数 (FMCN) 必须得到正确编程 (参见 “MC Flash 模式寄存器 ” on page 102)。

Note: 这个域定义了 1 个微秒的主时钟周期数，以确保一些必要的内部时序。

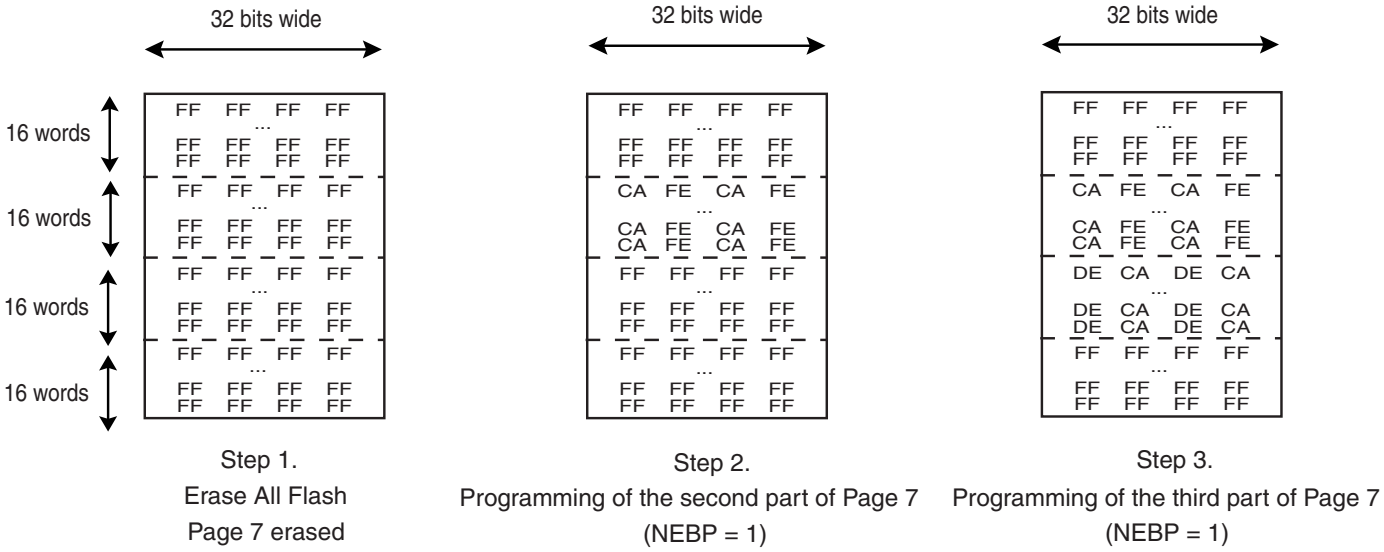
Flash 编程

有几个命令用于 Flash 编程。

Flash 技术决定了编程之前必须进行擦除。整个存储区可以同时进行擦除，也可以对页面自动进行擦除。具体操作是对 MC_FCR 寄存器写入命令之前清除 MC_FMR 寄存器的 NEBP。

置位MC_FMR寄存器的NEBP后，如果页面已经被擦除，则可以几个步骤进行编程(参见Figure 36)。

Figure 36. 页面部分编程的例子：



编程后，页(整个锁定区域)能上锁以防止其它写或擦除序列。使用 WPL 编程后锁定位将自动设定。

写入的数据存储于一个内部锁存缓冲器中。锁存缓冲器的大小由页的大小决定。锁存缓冲器在内部存储器区域地址空间中，重复的次数为页的数目。

Note: 不允许写 8 位或 16 位数据，因为可能会引起数据错误。

在编程命令写入 Flash 命令寄存器 MC_FCR 前将数据写入锁存缓冲器。序列如下：

- 对整个页中任意地址写入时，只允许 32 位访问。
- 页序号与编程命令写入 Flash 命令寄存器后编程启动。Flash 编程状态寄存器 (MC_FSR) 的 FRDY 位自动清零。
- 编程结束， FRDY 位上升。若通过设置 FRDY 使能中断，存储控制器中断线激活。

编程序列后， MC_FSR 寄存器能检测到两类错误：

- 编程错误：MC_FCR 寄存器中写入错误的关键字或一条无效的命令。
- 锁定错误：编程页属于锁定区域。事先必须将相应区域解锁。

擦除所有命令

若 MC_FCR 中擦除所有命令 (EA) 位写入，将擦除整个存储器。

只有当没有设置锁定位时允许擦除所有的操作。因此若至少有一个区域锁定， MC_FSR 中的 LOCKE 位上升并将命令取消。若 LOCKE 位已写入 1 则中断线上升。

编程结束， FRDY 位上升。若通过设置 FRDY 使能中断，存储控制器中断线激活。

编程序列后， MC_FSR 寄存器能检测到两类错误：

- 编程错误：MC_FCR 寄存器中写入错误的关键字或一条无效的命令。
- 锁定错误：至少有一个擦除区域有锁定保护。擦除命令无效且不会擦除任何页。事先必须将相应区域解锁。

锁定位保护

锁定位与内置 Flash 存储器中的几个页相关。它用来定义内置 Flash 存储器的锁定区域。它们防止写 / 擦除保护页。

出品时，器件可能有一些 Flash 锁定区域上锁。这些锁定区域用于默认应用。参见产品说明中关于默认的内置 Flash 映射部分。可通过擦除将锁定区域解锁，再对其编程。

锁定序列为：

- Flash 命令寄存器中必须写入下列值：
  (0x5A << 24) | (lockPageNumber << 8 & PAGEN) | SLB
  lockPageNumber 为锁定区域相应的页。
- 当锁定完成， FRDY 位上升。若通过设置 FRDY 使能中断，存储控制器中断线激活。

MC_FCR 寄存器中写入错误的关键字或一条无效的命令时，在编程序列后，MC_FSR 寄存器能检测到。

可清除预先设定的锁定位。这样可对锁定的区域进行擦除或编程。解锁序列为：

- Flash 命令寄存器中必须写入下列值：
  (0x5A << 24) | (lockPageNumber << 8 & PAGEN) | CLB
  lockPageNumber 为锁定区域相应的页。
- 当解锁完成， FRDY 位上升。若通过设置 FRDY 使能中断，存储控制器中断线激活。

MC_FCR 寄存器中写入错误的关键字或一条无效的命令时，在编程序列后，MC_FSR 寄存器能检测到。

解锁命令将锁定位编程为 1 ； MC_FSR 中相应位 LOCKSx 读 0。锁定命令将锁定位编程为 0 ； MC_FSR 中相应位 LOCKSx 读 1。

Note: 在读模式下，当执行锁定或解锁命令时允许访问 Flash。

通用功能 NVM 位

通用功能 NVM 位不与内置 Flash 存储器连接。这些通用功能位专门用于保护产品的其它部分。它们可独立置位 (激活) 或清除。参见产品说明中关于通用功能 NVM 位的说明。

激活序列为：

- 使用 SEL 命令对 Flash 命令寄存器写入，以启动设置通用功能位命令 (SGPB) ，并在 PAGEN 中设定激活通用功能位数目。
- 当该位置位， FRDY 位上升。若通过设置 FRDY 使能中断，存储控制器中断线激活。

编程序列后， MC_FSR 寄存器能检测到两类错误：

- 编程错误：MC_FCR 寄存器中写入错误的关键字或一条无效的命令。
- 若通用功能位数大于通用功能位总数，则命令无效。

可事先将通用功能 NVM 位中止，清除序列为：

- 使用 CGPB 命令对 Flash 命令寄存器写入，以启动清除通用功能位命令 (CGPB) ，并在 PAGEN 中设定清除通用功能位数目。
- 当清除完成， FRDY 位上升。若通过设置 FRDY 使能中断，存储控制器中断线激活。

编程序列后， MC_FSR 寄存器能检测到两类错误：

- 编程错误：MC_FCR 寄存器中写入错误的关键字或一条无效的命令。
- 若 PAGEN 中设置的通用功能位数大于通用功能位总数，则命令无效。

清除通用功能位命令将通用功能 NVM 位编程为 1 ； MC_FSR 中相应位 GPNVMx 读 0。设置通用功能位命令将通用功能 NVM 位编程为 0 ； MC_FSR 中相应位 GPNVMx 读 1。

Note: 在读模式下，当执行设置、清除或得到通用功能 NVM 位命令时允许访问 Flash。

安全位

安全位的目的是防止外部对内部总线系统的访问。JTAG、快速 Flash 编程及 Flash 串行测试接口特性禁用。置位后，该位只能通过外部硬件 ERASE 向芯片请求来复位。参见产品说明中关于控制 ERASE 引脚的说明。此时，整个存储器被擦除且所有锁定及通用功能 NVM 位被清除。只有在这些操作完成后，才会将 MC_FSR 中的安全位清除。

激活序列为：

- 对 Flash 命令寄存器写入以启动安全位命令 (SSB)。
- 当锁定完成， FRDY 位上升。若通过设置 FRDY 使能中断，存储控制器中断线激活。

当安全位激活， MC_FSR 中的 SECURITY 位置位。

内置 Flash 控制器 (EFC) 用户接口

内置 Flash 控制器的用户接口集成在存储控制器中。

基地址：0xFFFF FF00

Table 18. 内置 Flash 控制 (EFC) 寄存器映射

偏移	寄存器	名称	访问	复位状态
0x60	MC Flash 模式寄存器	MC_FMR	读 / 写	0x0
0x64	MC Flash 命令寄存器	MC_FCR	只写	–
0x68	MC Flash 状态寄存器	MC_FSR	只读	–
0x6C	保留	–	–	–

MC Flash 模式寄存器

寄存器名称 :MC_FMR

访问类型： 读 / 写

偏移： 0x60

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
FMCN							

15	14	13	12	11	10	9	8
–	–	–	–	–	–	FWS	

7	6	5	4	3	2	1	0
NEBP	–	–	–	PROGE	LOCKE	–	FRDY

- **FRDY: Flash 就绪中断使能**

0 = Flash 就绪不产生中断。

1 = Flash 就绪产生中断。

- **LOCKE: 锁定错误中断使能**

0 = 锁定错误不产生中断。

1 = 锁定错误产生中断。

- **PROGE: 编程错误中断使能**

0 = 编程错误不产生中断。

1 = 编程错误产生中断。

- **NEBP: 编程前无擦除**

0 = 编程前执行页擦除。

1 = 编程前无页擦除。

- **FWS: Flash 等待状态**

该域定义读、写操作时的等待状态数目：

FWS	读操作	写操作
0	1 周期	2 周期
1	2 周期	3 周期
2	3 周期	4 周期
3	4 周期	4 周期

- **FMCN: Flash 微秒周期数**

该域定义主时钟周期数，单位为微秒。

Warning: 只有在主机时钟周期超过 30 微秒时出现值 0。

Warning: 为保证对 Flash 存储器的有效操作， Flash 微秒周期数 (FMCN) **必须**编程正确。

MC Flash 命令寄存器

寄存器名称： MC_FCR

访问类型：只写

偏移： 0x64

31	30	29	28	27	26	25	24
KEY							

23	22	21	20	19	18	17	16
–	–	–	–	–	–	PAGEN	

15	14	13	12	11	10	9	8
PAGEN							

7	6	5	4	3	2	1	0
–	–	–	–	FCMD			

- **FCMD: Flash 命令**

该域定义 Flash 命令：

FCMD	操作
0000	无命令。 不拉高 MC_FSR 的编程错误标志。
0001	**写页命令 (WP):** 启动 PAGEN 域指定的页编程。
0010	**设置锁定位命令 (SLB):** 启动 PAGEN 域指定的锁定域的锁定位序列。
0011	**写页与锁定命令 (WPL):** 编程序列完成后，PAGEN 域指定的锁定域相关页的锁定序列自动出现。
0100	**清除锁定位命令 (CLB):** 启动 PAGEN 域指定的锁定域的清除锁定位序列。
1000	**擦除所有命令 (EA):** 启动擦除整个 Flash。 只要有一页锁定，该命令删除。
1011	**设置通用功能 NVM 位 (SGPB):** 激活 PAGEN 域指定的相应的通用功能 NVM 位数。
1101	**清除通用功能 NVM 位 (CGPB):** 禁用 PAGEN 域指定的相应的通用功能 NVM 位数。
1111	**设置安全位命令 (SSB):** 设置安全位。
其它	保留。 拉高 MC_FSR 的编程错误标志。

- **PAGEN: 页数**

命令	PAGEN 说明
写页命令	PAGEN 定义写页数目。
写页与锁定命令	PAGEN 定义写页及相关锁定域的数目。
擦除所有命令	该域无意义。
设置 / 清除锁定位命令	PAGEN 定义锁定或解锁的锁定域页数目。
设置 / 清除通用功能 NVM 位命令	PAGEN 定义通用功能位数目。
设置安全位命令	该域无意义。

Note: 除上述命令外，所有 PAGEN 中的未使用位无意义。

- **KEY: 写保护键**

该域应写入 0x5A 以使能由寄存器中的相应位定义的命令。若写入其它值，不会执行写操作且不会启动其它操作。

MC Flash 状态寄存器

寄存器名称 :MC_FSR

访问类型： 只读

偏移： 0x68

31	30	29	28	27	26	25	24
LOCKS15	LOCKS14	LOCKS13	LOCKS12	LOCKS11	LOCKS10	LOCKS9	LOCKS8
23	22	21	20	19	18	17	16
LOCKS7	LOCKS6	LOCKS5	LOCKS4	LOCKS3	LOCKS2	LOCKS1	LOCKS0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	GPNVM1	GPNVM0
7	6	5	4	3	2	1	0
–	–	–	SECURITY	PROGE	LOCKE	–	FRDY

- **FRDY: Flash 就绪状态**

0 = 内置 Flash 控制器忙且在运行新命令前需要等待。

1 = 内置 Flash 控制器就绪以等待新命令。

- **LOCKE: 锁定错误状态**

0 = 在最后读 MC_FSR 后没有对锁定的锁定区域编程。

1 = 在最后读 MC_FSR 后至少对一个锁定的锁定区域编程。

- **PROGE: 编程错误状态**

0 = 对 MC_FCR 写入时无无效命令或错误关键字。

1 = 对 MC_FCR 写入时有无效命令或错误关键字。

- **SECURITY: 安全位状态**

0 = 安全位禁用。

1 = 安全位激活。

- **GPNVMx: 通用功能 NVM 位状态**

0 = 相应的通用功能 NVM 位禁用。

1 = 相应的通用功能 NVM 位激活。

- **LOCKSx: 锁定域 x 锁定状态**

0 = 相应锁定域未锁定。

1 = 相应锁定域锁定。

快速 Flash 编程接口 (FFPI)

概述

快速 Flash 编程接口对使用标准群编程的大容量编程提供两种解决方式：并行或串行。并行接口是全握手的且将器件认为是标准 EEPROM。此外，并行协议还向所有内置 Flash 提供优化的访问方式。串行接口使用标准IEEE 1149.1 JTAG协议，向所有内置Flash提供优化的访问方式。

尽管快速 Flash 编程模式是专门针对大容量编程模式，但该模式并不是为 in-situ 编程设计的。

并行快速 Flash 编程

器件配置

在快速 Flash 编程模式下，器件在专门的测试模式下。只有特定的引脚有意义，其它引脚不能连接。

Figure 37. 并行编程接口

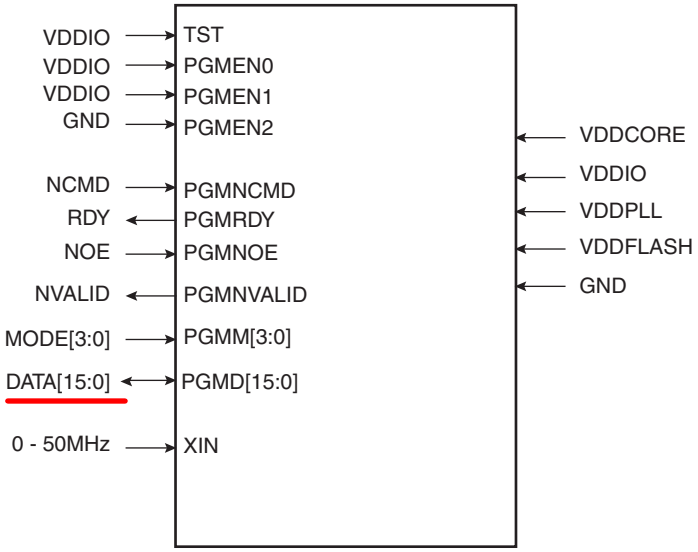


Table 19. 信号说明列表

信号名称	功能	类型	有效电平	注释
电源				
VDDFLASH	Flash 电源	电源		
VDDIO	I/O 线电源	电源		
VDDCORE	内核电源	电源		
VDDPLL	备用 I/O 线电源	电源		
GND	地	地		
时钟				
XIN	主时钟输入。 该输入可与 GND 连接。此时，器件时钟由内部 RC 振荡器提供。	输入		32KHz 到 50MHz
测试				
TST	测试模式选择	输入	高	必须与 VDDIO 连接
PGMEN0	测试模式选择	输入	高	必须与 VDDIO 连接
PGMEN1	测试模式选择	输入	高	必须与 VDDIO 连接
PGMEN2	测试模式选择	输入	低	必须与 GND 连接

Table 19. 信号说明列表

信号名称	功能	类型	有效电平	注释
PIO				
PGMNCMD	有效命令可用	输入	低	复位时上拉输入
PGMRDY	0：器件忙 1：器件可开始执行新命令	输出	高	复位时上拉输入
PGMNOE	输出使能 (高有效)	输入	低	复位时上拉输入
PGMNVALID	0：DATA[15:0] 为输入模式 1：DATA[15:0] 为输出模式	输出	低	复位时上拉输入
PGMM[3:0]	中断 DATA 类型 (见 Table 20)	输入		复位时上拉输入
PGMD[15:0]	双向数据总线	输入 / 输出		复位时上拉输入

信号名称

由 MODE 设置确定， DATA 锁存在不同的内部寄存器中。

Table 20. 模式代码

MODE[3:0]	符号	数据
0000	CMDE	命令寄存器
0001	ADDR0	地址寄存器，低位在先
0010	ADDR1	
0011	ADDR2	
0100	ADDR2	地址寄存器，高位在先
0101	DATA	数据寄存器
默认	IDLE	无寄存器

当 MODE 等于 CMDE，则新命令 (由 DATA[15:0] 信号选通) 存于命令寄存器中。

Table 21. 命令位代码

DATA[15:0]	符号	命令执行
0x0011	READ	读 Flash
0x0012	WP	写 Flash 页
0x0022	WPL	写页并锁定 Flash
0x0032	EWP	擦除并写页
0x0042	EWPL	擦除后写页并锁定
0x0013	EA	擦除所有
0x0014	SLB	设置锁定位
0x0024	CLB	清除锁定位
0x0015	GLB	得到锁定位
0x0034	SFB	设置通用功能 NVM 位
0x0044	CFB	清除通用功能 NVM 位
0x0025	GFB	得到通用功能 NVM 位

Table 21. 命令位代码

DATA[15:0]	符号	命令执行
0x0054	SSE	设置安全位
0x0035	GSE	得到安全位
0x001E	GVE	得到版本

进入编程模式

通过下面算法使器件进入并行编程模式：

- 提供 GND、 VDDIO、 VDDCORE、 VDDFLASH 及 VDDPLL。
- 若外部时钟有效，提供 XIN 时钟为 T_{POR_RESET}。
- 等待 T_{POR_RESET}。
- 启动读或写握手。

Note: 复位后，器件由内部 RC 振荡器通过时钟。清除 RDY 信号前，若外部时钟 (> 32 kHz) 与 XIN 连接，则器件切换到外部时钟。此外，不考虑 XIN 输入。 XIN 上更高频率加速程序握手。

程序握手

握手是对读、写操作而定义的。当器件准备启动一个新的操作 (RDY 信号设置)，程序通过清除 NCMD 信号启动握手。当 NCMD 及 RDY 为高，实现握手。

写握手

详见 Figure 38 及 Table 22。

Figure 38. 写序列的并行编程时序

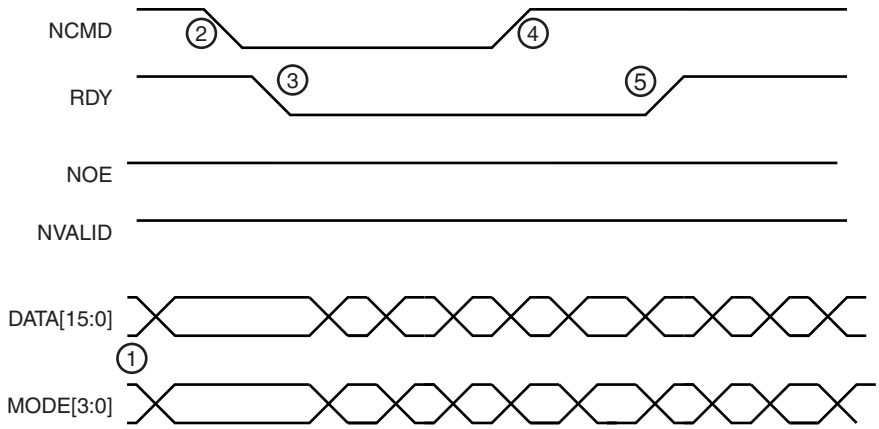


Table 22. 写握手

步骤	程序员操作	器件行为	数据 I/O
1	设置 MODE 及 DATA 信号	等待 NCMD 为低	输入
2	清除 NCMD 信号	锁存 MODE 与 DATA	输入
3	等待 RDY 为低	清除 RDY 信号	输入
4	释放 MODE 与 DATA 信号	执行命令并轮询 NCMD 为高	输入
5	设置 NCMD 信号	执行命令并轮询 NCMD 为高	输入
6	等待 RDY 为高	设置 RDY	输入

读握手

详见 Figure 39 与 Table 23。

Figure 39. 读序列的并行编程时序

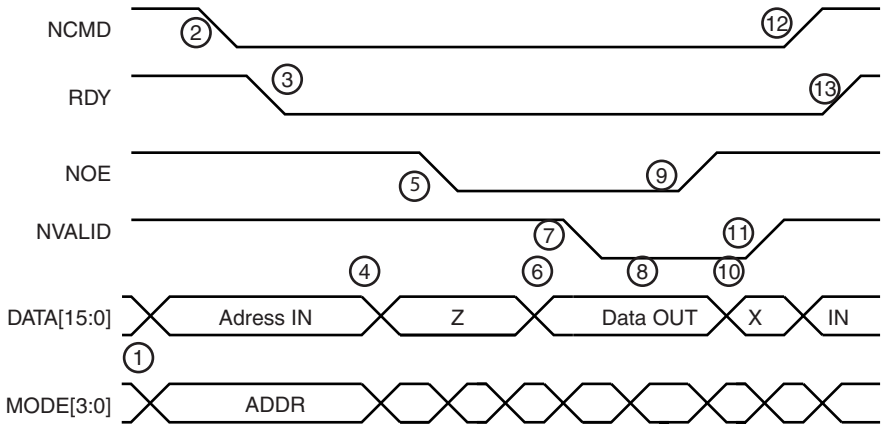


Table 23. 读握手

步骤	程序员操作	器件行为	数据 I/O
1	设置 MODE 及 DATA 信号	等待 NCMD 为低	输入
2	清除 NCMD 信号	锁存 MODE 与 DATA	输入
3	等待 RDY 为低	清除 RDY 信号	输入
4	设置 DATA 信号为三态	等待 NOE 为低	输入
5	清除 NOE 信号		三态
6	等待 NVALID 为低	将 DATA 设置为输出模式并输出 flash 中内容	输出
7		清除 NVALID 信号	输出
8	读 DATA 总线值	等待 NOE 为高	输出
9	设置 NOE 信号		输出
10	等待 NVALID 为高	将 DATA 设置为输入模式	X
11	将 DATA 设置为输出模式	设置 NVALID 信号	输入
12	设置 NCMD 信号	等待 NCMD 为高	输入
13	等待 RDY 为高	设置 RDY 信号	输入

器件操作

Flash 存储器中有几条命令有效。命令小结见 Table 21 on page 109。程序员通过并行接口运行几个读 / 写握手序列驱动每条命令。

当执行新命令时，前一条命令自动完成。因此，在写命令后加读命令将自动刷新载入缓冲器。

Flash 读命令

该命令用来读 Flash 存储器内容。读命令可在存储器中任意有效地址开始，并对连续读进行了优化。内部地址缓冲器自动增加将建立读握手。

Table 24. 读命令

步骤	握手序列	MODE[3:0]	DATA[15:0]
1	写握手	CMDE	READ
2	写握手	ADDR0	32 位存储器地址首字节
3	写握手	ADDR1	32 位 Flash 地址
4	写握手	ADDR2	32 位 Flash 地址
5	写握手	ADDR3	32 位 Flash 地址尾字节
6	读握手	DATA	* 存储器地址 ++
7	读握手	DATA	* 存储器地址 ++
...	...	...	...
n	写握手	ADDR0	32 位存储器地址首字节
n+1	写握手	ADDR1	32 位 Flash 地址
n+2	写握手	ADDR2	32 位 Flash 地址
n+3	写握手	ADDR3	32 位 Flash 地址尾字节
n+4	读握手	DATA	* 存储器地址 ++
n+5	读握手	DATA	* 存储器地址 ++
...	...	...	...

Flash 写命令

该命令用来写 Flash 内容。

Flash存储器分在几个页中。要写的数据存储在对应于Flash存储器页的载入缓冲器中。载入缓冲器自动刷新 Flash：

- 在访问其它页前
- 当新命令有效 (MODE = CMDE)

写页命令 **(WP)** 对连续写进行了优化 。内部地址缓冲自动增加将建立写握手。

Table 25. 写命令

步骤	握手序列	MODE[3:0]	DATA[15:0]
1	写握手	CMDE	WP 或 WPL 或 EWP 或 EWPL
2	写握手	ADDR0	32 位存储器地址首字节
3	写握手	ADDR1	32 位 Flash 地址
4	写握手	ADDR2	32 位 Flash 地址
5	写握手	ADDR3	32 位 Flash 地址尾字节
6	写握手	DATA	* 存储器地址 ++
7	写握手	DATA	* 存储器地址 ++
...	...	...	...
n	写握手	ADDR0	32 位存储器地址首字节
n+1	写握手	ADDR1	32 位 Flash 地址
n+2	写握手	ADDR2	32 位 Flash 地址

Table 25. 写命令

步骤	握手序列	MODE[3:0]	DATA[15:0]
n+3	写握手	ADDR3	32 位 Flash 地址尾字节
n+4	写握手	DATA	* 存储器地址 ++
n+5	写握手	DATA	* 存储器地址 ++
...	...	...	...

Flash 命令**写页并锁定 (WPL)** 与 Flash 写命令等价。但是，锁定位在 Flash 写操作结束后自动设置。由于锁定域由几页组成，程序员使用 Flash 写命令对锁定域的首页写入；使用 Flash 写与锁定命令对锁定域的尾页写入。

Flash 命令**擦除页与写 (EWP)** 与 Flash 写命令等价。但是，在对载入缓冲器编程前将页擦除。

Flash 命令**擦除页与写锁定 (EWPL)** 结合 EWP 与 WPL 命令。

Flash 全擦除命令

该命令用于擦除 Flash 存储器。

使用 CLB 命令进行全擦除时必须将所有锁定域解锁。否则，放弃擦除命令并不执行页擦除。

Table 26. 全擦除命令

步骤	握手序列	MODE[3:0]	DATA[15:0]
1	写握手	CMDE	EA
2	写握手	DATA	0

Flash 锁定命令

使用 WPL 或 EWPL 命令设置锁定位。还可使用**设置锁定**命令 **(SLB)** 来设置。使用该命令，将激活几个锁定位。位屏蔽作为命令的变量。当位 0 屏蔽设置，则第一个锁定位激活。

同样，**清除锁定**命令 **(CLB)** 用来清除锁定位。所有锁定位可通过 EA 命令清除。

Table 27. 设置并清除锁定位命令

步骤	握手序列	MODE[3:0]	DATA[15:0]
1	写握手	CMDE	SLB 或 CLB
2	写握手	DATA	位屏蔽

使用**得到锁定位**命令 **(GLB)** 读取锁定位。当位 n 屏蔽设置，第 n 位锁定位激活。

Table 28. 得到锁定位命令

步骤	握手序列	MODE[3:0]	DATA[15:0]
1	写握手	CMDE	GLB
2	读握手	DATA	锁定位屏蔽状态 0 = 锁定位清除 1 = 锁定位置位

Flash 通用功能 NVM 命令

使用**设置熔丝位**命令 **(SFB)** 来设置通用功能 NVM 位 (GP NVM 位)。该命令同时激活 GP NVM 位。位屏蔽作为命令的变量。当位 0 屏蔽设置，则第一个 GP NVM 位激活。

同样，**清除熔丝位**命令 **(CFB)** 用来清除通用功能 NVM 位。所有的通用功能 NVM 位 可通过 EA 命令清除。当相应的值置 1，通用功能 NVM 位失效。

Table 29. 设置 / 清除 GP NVM 命令

步骤	握手序列	MODE[3:0]	DATA[15:0]
1	写握手	CMDE	SFB 或 CFB
2	写握手	DATA	GP NVM 位模式值

使用**得到熔丝位**命令 **(GFB)** 读取通用功能 NVM 位。当位 n 屏蔽设置 ,第 n 位 GP NVM 位激活。.

Table 30. 得到 GP NVM 位命令

步骤	握手序列	MODE[3:0]	DATA[15:0]
1	写握手	CMDE	GFB
2	读握手	DATA	GP NVM 位屏蔽状态 0 = GP NVM 位清除 1 = GP NVM 位置位

Flash 安全位命令

通过**设置安全位**命令 **(SSE)** 对安全位设置。若安全位激活，快速 Flash 编程将禁用。不能运行其它命令。当 Flash 中内容被擦除，可通过擦除引脚来擦除安全位。

Table 31. 设置安全位命令

步骤	握手序列	MODE[3:0]	DATA[15:0]
1	写握手	CMDE	SSE
2	写握手	DATA	0

得到版本命令

得到版本 (GVE) 命令找到 FFPI 接口版本。

Table 32. 得到版本命令

步骤	握手序列	MODE[3:0]	DATA[15:0]
1	写握手	CMDE	GVE
2	写握手	DATA	版本

串行快速 Flash 编程

串行快速 Flash 编程接口基于 IEEE Std. 1149.1 “ 标准测试访问端口与边界扫描架构 ”。术语及 TAP 控制器状态说明可参见该标准。

该模式下，器件内置 Flash 的数据读写通过 JTAG 接口。

器件配置

串行快速 Flash 编程模式下，器件在专门的测试模式下。只有部分引脚有意义，其它引脚不许连接。

Figure 40. 串行编程

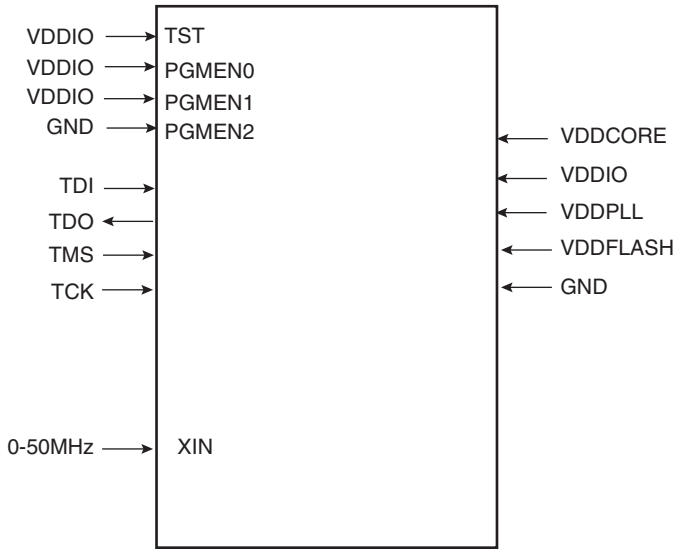


Table 33. 信号说明列表

信号名称	功能	类型	有效电平	注释
电源				
VDDFLASH	Flash 电源	电源		
VDDIO	I/O 线电源	电源		
VDDCORE	内核电源	电源		
VDDPLL	备用 I/O 线电源	电源		
GND	地	地		
时钟				
XIN	主时钟输入。 该输入可与 GND 连接。此时，器件时钟由内部 RC 振荡器提供。	输入		32 kHz 到 50 MHz

Table 33. 信号说明列表

信号名称	功能	类型	有效电平	注释
测试				
TST	测试模式选择	输入	高	必须与 VDDIO 连接
PGMEN0	测试模式选择	输入	高	必须与 VDDIO 连接
PGMEN1	测试模式选择	输入	高	必须与 VDDIO 连接
PGMEN2	测试模式选择	输入	低	必须与 GND 连接
JTAG				
TCK	JTAG TCK	输入	-	复位时上拉输入
TDI	JTAG 测试数据输入	输入	-	复位时上拉输入
TDO	JTAG 测试数据输出	输出	-	
TMS	JTAG 测试模式选择	输入	-	复位时上拉输入

进入串行编程模式

通过下面算法使器件进入串行编程模式：

- 提供 GND、 VDDIO、 VDDCORE、 VDDFLASH 与 VDDPLL。
- 若外部时钟有效，提供 XIN 时钟为 T_{POR_RESET} + 32(T_{SCLK})。
- 等待 T_{POR_RESET}。
- 设置 TMS 为 5 TCK 脉冲复位 TAP 控制器。
- 不通过运行 - 测试空闲状态将 0x2 移入 IR 寄存器 (IR 为 4 位，低位在先) 。
- 不通过运行 - 测试空闲状态将 0x2 移入 DR 寄存器 (DR 为 4 位，低位在先) 。
- 不通过运行 - 测试空闲状态将 0xC 移入 IR 寄存器 (IR 为 4 位，低位在先) 。

Note: 复位后，器件由内部 RC 振荡器通过时钟。清除 RDY 信号前，若外部时钟 (> 32 kHz) 与 XIN 连接，则器件切换到外部时钟。此外，不考虑 XIN 输入。 XIN 上更高频率加速程序握手。

Table 34. 复位 TAP 控制器并进入选择 -DR- 扫描

TDI	TMS	TAP 控制器状态
X	1	
X	1	
X	1	
X	1	
X	1	测试 - 逻辑复位
X	0	运行 - 测试 / 空闲
Xt	1	选择 -DR- 扫描

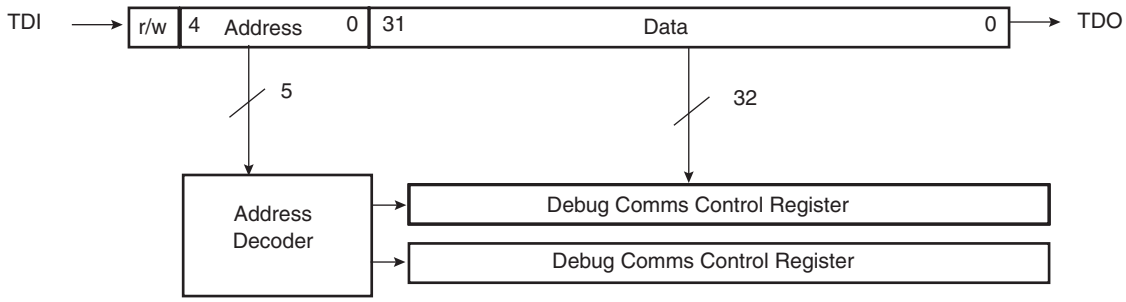
读 / 写握手

有两个寄存器通过 JTAG 接口访问：

- 调试通信控制寄存器：DCCR
- 调试通信数据寄存器：DCDR

通过 TAP 38 位 DR 寄存器访问这些寄存器， 38 位 DR 寄存器包括 32 位数据域， 5 位地址域及一个读 / 写位。写入数据扫描到由 5 位地址域确定的 32 位数据域中，而读 / 写位写入 1。寄存器读取时扫描地址域且读 / 写位写入 0，通过 UPDATE-DR TAP 状态，然后扫描输出数据。32 位数据域忽略。

Figure 41. TAP 8 位 DR 寄存器



TAP 控制器进入 UPDATE-DR 状态时出现读或写。

- 调试通信控制寄存器地址为 0x04。
- 调试通信数据寄存器地址为 0x05。

调试通信控制寄存器只读且允许处理器与调试器同步握手。

- 位 1 (W) : 指示程序是否可通过调试通信数据寄存器读取数据。若器件忙 W = 0，则程序轮询直到 W = 1。
- 位 0 (R) : 指示程序是否可通过调试通信数据寄存器发送数据。若 R = 1，表示前面位置的数据未处理，程序必须等待。

器件操作

Flash 存储器中有几条命令有效。命令小结见 Table 21 on page 109。程序员通过串行接口对调试通信寄存器进行读写。

Flash 读命令

该命令用来读 Flash 存储器内容。存储器映射通过该命令访问。存储器为字阵列 (32 位宽)。读命令可在存储器中任意有效地址开始，**地址必须字对齐**。地址自动增加。

Table 35. 读命令

读 / 写	DR 数据
写	(读的字数) << 16 | READ
写	地址
读	存储器 [地址]
读	存储器 [地址 +4]
...	...
读	存储器 [地址 +(读的字数 - 1)* 4]

Flash 写命令

该命令用来写 Flash 内容。传输地址必须是存储器中有效地址。

Flash存储器分在几个页中。要写的数据存储在对应于Flash存储器页的载入缓冲器中。载入缓冲器自动刷新 Flash :

- 在访问其它页前
- 字传输结束时

写页命令 **(WP)** 对连续写进行了优化 。内部地址缓冲自动增加将建立写握手。

Table 36. 写命令

读 / 写	DR 数据
写	(写字数) << 16	(WP 或 WPL 或 EWP 或 EWPL)
写	地址
写	存储器 [地址]
写	存储器 [地址 +4]
写	存储器 [地址 +8]
写	存储器 [地址 +(写字数 - 1)* 4]

Flash 命令**写页并锁定 (WPL)** 与 Flash 写命令等价。但是 ,锁定位在 Flash 写操作结束后自动设置。由于锁定域由几页组成 ,程序员使用 Flash 写命令对锁定域的首页写入；使用 Flash 写与锁定命令对锁定域的尾页写入。

Flash 命令**擦除页与写 (EWP)** 与 Flash 写命令等价。但是，在对载入缓冲器编程前将页擦除。

Flash 命令**擦除页与写锁定 (EWPL)** 结合 EWP 与 WPL 命令。

Flash 全擦除命令

该命令用于擦除 Flash 存储器。

使用 CLB 命令进行全擦除时必须将所有锁定域解锁。

Table 37. 全擦除命令

读 / 写	DR 数据
写	EA

Flash 锁定命令

使用 WPL 或 EWPL 命令设置锁定位。还可使用**设置锁定**命令 **(SLB)** 来设置。使用该命令，将激活几个锁定位。位屏蔽作为命令的变量。当位 0 屏蔽设置，则第一个锁定位激活。

同样，**清除锁定**命令 **(CLB)** 用来清除锁定位。所有锁定位可通过 EA 命令清除。

Table 38. 设置与清除锁定位命令

读 / 写	DR 数据
写	SLB 或 CLB
写	位屏蔽

使用**得到锁定位**命令 **(GLB)** 读取锁定位。当位屏蔽返回，相应锁定位激活。

Table 39. 得到锁定位命令

读 / 写	DR 数据
写	GLB
读	位屏蔽

Flash 通用功能 NVM 命令

使用**设置熔丝位**命令 **(SFB)** 来设置通用功能 NVM 位 (GP NVM 位)。该命令可同时激活几个 GP NVM 位。位 0 的位屏蔽对应第一个熔丝位，依此类推。

同样，**清除熔丝位**命令 **(CFB)** 用来清除通用功能 NVM 位。所有的通用功能 NVM 位 可通过 EA 命令清除。

Table 40. 设置与清除通用功能 NVM 位命令

读 / 写	DR 数据
写	SFB 或 CFB
写	位屏蔽

使用**得到熔丝位**命令 **(GFB)** 读取通用功能 NVM 位。当位屏蔽返回，相应熔丝位置位。

Table 41. 得到通用功能 NVM 位命令

读 / 写	DR 数据
写	GFB
读	位屏蔽

Flash 安全位命令

通过**设置安全位**命令 **(SSE)** 对安全位设置。若安全位激活，快速 Flash 编程将禁用。不能运行其它命令。当 Flash 中内容被擦除，可通过擦除引脚来擦除安全位。

Table 42. 设置安全位命令

读 / 写	DR 数据
写	SSE

得到版本命令

得到版本 (GVE) 命令找到 FFPI 接口版本。

Table 43. 得到版本命令

读 / 写	DR 数据
写	GVE
读	版本

外设数据控制器 (PDC)

概述

外设数据控制器 (PDC) 在诸如 UART、 USART、 SSC、 SPI、 MCI 等片上外设与片内或片外存储器间传输数据。使用外设数据控制器避免处理器干涉并减去了处理器中断处理开销。这显著减少了数据传输所需时钟周期数并提高了微控制器性能，使其更加高效。

PDC 通道是成对的，每对对应一个指定的外设。其中一条通道负责接收数据，另一条通道负责发送数据。

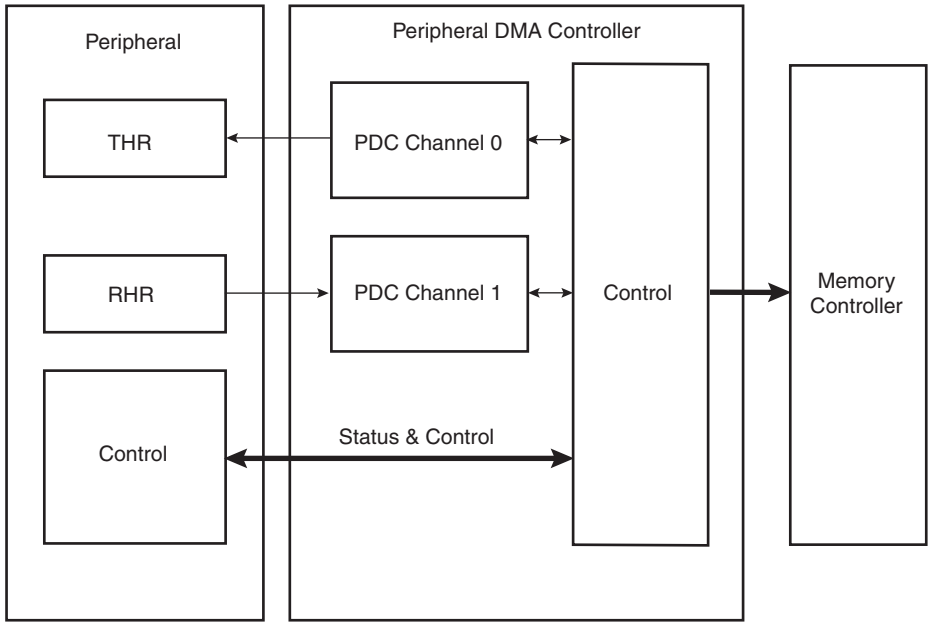
PDC 通道的用户接口集成在每个外设的存储器空间中，它包括：

- 32 位存储器指针寄存器
- 16 位传输计数寄存器
- 32 位下一存储器指针寄存器
- 16 位下一传输计数寄存器

外设通过发送与接收信号触发 PDC 传输。当传输编程数据时，相应的外设产生传输中断结束传输。

方框图

Figure 42. 方框图



功能说明

配置

PDC 通道用户接口使能用户对每条通道数据传输的配置与控制。PDC 通道用户接口集成在与其相关的外设用户接口中 (偏移 0x100)。

每个外设包括 4 个 32 位指针寄存器 (RPR、 RNPR、 TPR 及 TNPR) 与 4 个 16 位计数寄存器 (RCR、 RNCR、 TCR 及 TNCR)。

内部 16 位传输计数寄存器配置缓冲器大小 (传输数目) , 且可在任意时刻读取通道剩余传输数据数目。

存储器基地址通过定义访问存储器首地址配置为一个 32 位存储器指针。可在任意时刻读取下一个传输的地址及待传输数据数目。PDC 有专用状态寄存器用来指示每条通道传输是否使能。每条通道状态位于外设状态寄存器中。通过设置 PDC 传输控制寄存器的 TXTEN/TXTDIS 与 RXTEN/RXTDIS 可使能 / 禁用传输。这些控制位保证安全使能读取指针与计数寄存器。

PDC 发送到外设状态寄存器中的状态标志可见 (ENDRX、 ENDTX、 RXBUFF 及 TXBUFE)。

当 PERIPH_RCR 寄存器为零时 ENDRX 标志置位。

当 PERIPH_RCR 与 PERIPH_RNCR 寄存器均为零时 RXBUFF 标志置位。

当 PERIPH_TCR 寄存器为零时 ENDTX 标志置位。

当 PERIPH_TCR 与 PERIPH_TNCR 寄存器均为零时 TXBUFE 标志置位。

这些状态寄存器说明见外设状态寄存器。

存储器指针

每个外设通过一个接收数据通道及一个发送数据通道与 PDC 连接。每条通道有一个内部 32 位存储器指针。每个存储器指针可指向存储器空间任意位置(片上存储器或外部总线接口存储器)。

根据传输类型的不同 (字节、半字或字) , 存储器指针以 1、 2 或 4 增加。

若 PDC 工作时存储器指针重编程 , 传输地址改变且 PDC 使用新地址执行传输。

传输计数器

每条通道有一个内部 16 位传输计数器用来计算相关通道已传输的块。这些计数器在每次数据传输后自减。当计数器值为零时 , 传输完成 , PDC 停止传输数据。

若下一个计数寄存器值为零 , 当激活相关外设结束标志时 PDC 禁用触发。

若 PDC 工作时计数器重编程 , 传输数目更新 , PDC 传输由新值开始计数。

将下一个计数 / 指针寄存器与缓冲器链接。计数器在每次数据传输后自减但当传输计数器值为零时下一个计数 / 指针寄存器值载入计数 / 指针寄存器以重新使能触发。

对每条通道两个状态位指示当前缓冲器结束 (ENDRX、 ENTX) 及当前与下一个缓冲器结束 (RXBUFF、 TXBUFE)。这些位直接映射到外设状态寄存器且可触发 AIC 中断请求。

当某一个计数寄存器 (计数器或下一个计数寄存器) 写入时 , 外设结束标志自动清除。

注意 : 当下一个计数寄存器载入计数寄存器时 , 它将设置为零。

数据传输

外设使用发送 (TXRDY) 与接收 (RXRDY) 信号来触发 PDC 传输。

当外设收到外部符号 , 它向 PDC 发送接收就绪信号。当访问被允许 , PDC 开始读取外设接收保持寄存器 (RHR) 并触发存储器写操作。

每次传输后 , 相关 PDC 存储器指针自加 , 传输剩余数目自减。当传输完成 , 向外设发送信号并停止传输。

发送传输时步骤相同。

PDC 传输请求优先权

外设数据控制器根据每种产品固定的优先级来处理传输请求。具体优先级见产品手册。

若在同一外设上同时出现相同类型 (接收或发送) 请求，优先级由外设序号确定。

若请求不是同时出现，它们按照出现次序处理。先处理接收请求然后处理发送请求。

外设数据控制器 (PDC) 用户接口

Table 44. 外设数据控制器 (PDC) 寄存器映射

偏移	寄存器	寄存器名称	访问类型	复位
0x100	接收指针寄存器	PERIPH[(1)]_RPR	读 / 写	0x0
0x104	接收计数寄存器	PERIPH_RCR	读 / 写	0x0
0x108	发送指针寄存器	PERIPH_TPR	读 / 写	0x0
0x10C	发送计数寄存器	PERIPH_TCR	读 / 写	0x0
0x110	接收下一指针寄存器	PERIPH_RNPR	读 / 写	0x0
0x114	接收下一计数寄存器	PERIPH_RNCR	读 / 写	0x0
0x118	发送下一指针寄存器	PERIPH_TNPR	读 / 写	0x0
0x11C	发送下一计数寄存器	PERIPH_TNCR	读 / 写	0x0
0x120	PDC 传输控制寄存器	PERIPH_PTCR	只写	-
0x124	PDC 传输状态寄存器	PERIPH_PTSR	只读	0x0

Note: 1. PERIPH：十个寄存器映射在外设存储器空间偏移相同。用户可根据功能与外设需求 (DBGU、 USART、 SSC、 SPI、 MCI 等) 对其进定义。

PDC 接收指针寄存器

寄存器名称： PERIPH_RPR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
RXPTR							
23	22	21	20	19	18	17	16
RXPTR							
15	14	13	12	11	10	9	8
RXPTR							
7	6	5	4	3	2	1	0
RXPTR							

- **RXPTR: 接收指针地址**

下一个接收传输地址。

PDC 接收计数寄存器

寄存器名称： PERIPH_RCR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
--							
23	22	21	20	19	18	17	16
--							
15	14	13	12	11	10	9	8
RXCTR							
7	6	5	4	3	2	1	0
RXCTR							

- **RXCTR: 接收计数器值**

接收的传输数据数目。

PDC 发送指针寄存器

寄存器名称： PERIPH_TPR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
TXPTR							
23	22	21	20	19	18	17	16
TXPTR							
15	14	13	12	11	10	9	8
TXPTR							
7	6	5	4	3	2	1	0
TXPTR							

- **TXPTR: 发送指针地址**

发送缓冲器地址。

PDC 发送计数寄存器

寄存器名称： PERIPH_TCR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
--							
23	22	21	20	19	18	17	16
--							
15	14	13	12	11	10	9	8
TXCTR							
7	6	5	4	3	2	1	0
TXCTR							

- **TXCTR: 发送计数值**

TXCTR 为将发送数据数目。为零时外设数据传输停止。

PDC 接收下一指针寄存器

寄存器名称： PERIPH_RNPR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
RXNPTR							
23	22	21	20	19	18	17	16
RXNPTR							
15	14	13	12	11	10	9	8
RXNPTR							
7	6	5	4	3	2	1	0
RXNPTR							

- **RXNPTR: 接收下一指针地址**

RXNPTR 为当前缓冲器满时下一个填充接收数据的缓冲器地址。

PDC 接收下一计数寄存器

寄存器名称： PERIPH_RNCR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
--							
23	22	21	20	19	18	17	16
--							
15	14	13	12	11	10	9	8
RXNCR							
7	6	5	4	3	2	1	0
RXNCR							

- **RXNCR: 接收下一计数值**

RXNCR 为下一个接收缓冲器大小。

PDC 发送下一指针寄存器

寄存器名称： PERIPH_TNPR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
TXNPTR							
23	22	21	20	19	18	17	16
TXNPTR							
15	14	13	12	11	10	9	8
TXNPTR							
7	6	5	4	3	2	1	0
TXNPTR							

- **TXNPTR: 发送下一指针地址**

TXNPTR 为当前缓冲器空时下一个要发送的缓冲器地址。

PDC 发送下一计数寄存器

寄存器名称： PERIPH_TNCR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
--							
23	22	21	20	19	18	17	16
--							
15	14	13	12	11	10	9	8
TXNCR							
7	6	5	4	3	2	1	0
TXNCR							

- **TXNCR: 发送下一计数值**

TXNCR 为下一发送缓冲器大小。

PDC 传输控制寄存器

寄存器名称： PERIPH_PTCR

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXTDIS	TXTEN
7	6	5	4	3	2	1	0
–	–	–	–	–	–	RXTDIS	RXTEN

- **RXTEN: 接收传输使能**

0 = 无效。

1 = 若 RXTDIS 未置位，使能接收 PDC 传输请求。

- **RXTDIS: 接收传输禁用**

0 = 无效。

1 = 禁用接收 PDC 传输请求。

- **TXTEN: 发送传输使能**

0 = 无效。

1 = 使能发送 PDC 传输请求。

- **TXTDIS: 发送传输禁用**

0 = 无效。

1 = 禁用发送 PDC 传输请求。

PDC 传输状态寄存器

寄存器名称： PERIPH_PTSR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	TXTEN
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	RXTEN

- **RXTEN: 接收传输使能**

0 = 禁用接收 PDC 传输请求。

1 = 使能接收 PDC 传输请求。

- **TXTEN: 发送传输使能**

0 = 禁用发送 PDC 传输请求。

1 = 使能发送 PDC 传输请求。

高级中断控制器 (AIC)

概述

高级中断控制器 (AIC) 是有 8 个优先级，独立可屏蔽的向量中断控制器，最多可处理 32 个中断源。它的设计充分减少了处理内部与外部中断中的软件与实时开销。

AIC驱动ARM处理器的nFIQ (快速中断请求)与nIRQ (标准中断请求)输入。AIC输入即可为内部外设中断也可为来自产品引脚的外部中断。

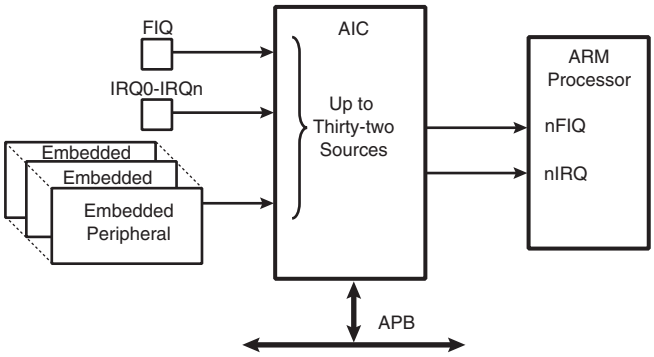
8优先级控制器允许用户对每个中断源定义其优先级，并允许高优先级中断打断低优先级中断来处理。

内部中断源可编程为电压敏感或边沿触发。外部中断源可编程为正边沿或负边沿触发，或高电平或低电平敏感。

快速强制特性将内部或外部中断源改变为一个快速中断。

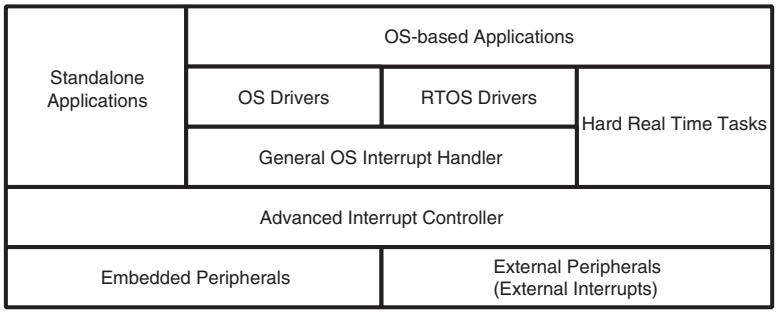
方框图

Figure 43. 方框图



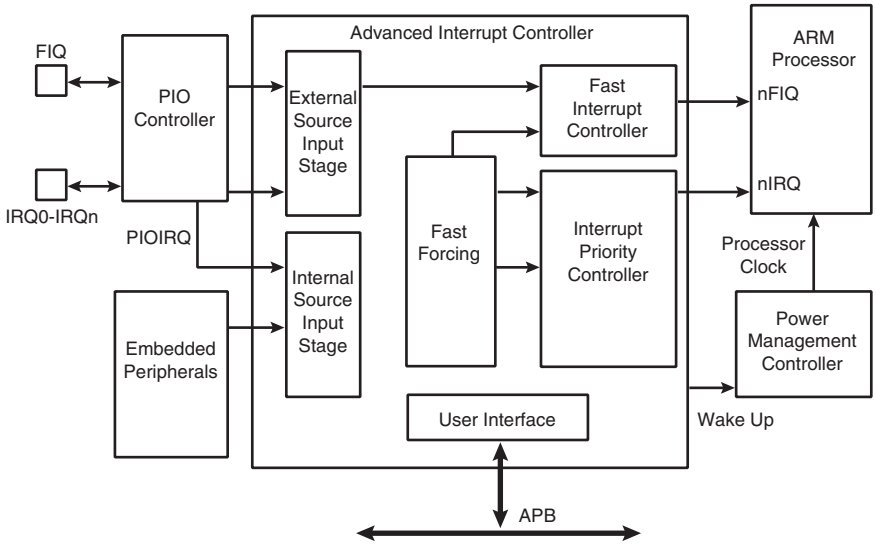
应用框图

Figure 44. 应用框图说明



AIC 详细框图

Figure 45. AIC 详细框图



I/O 线说明

Table 45. I/O 线说明

引脚名称	引脚使能	类型
FIQ	快速中断	输入
IRQ0 - IRQn	中断 0 - 中断 n	输入

附属产品

I/O 线

中断信号 FIQ 与 IRQ0 到 IRQn 一般通过 PIO 控制器复用。取决于所用产品的 PIO 控制器特性，引脚必须根据其分配的中断功能进行编程。当 PIO 控制器输入路径透明时不可用。

电源管理

高级中断控制器是连续时钟驱动的。电源管理控制器对高级中断控制器无效。

高级中断控制器输出出现 (nIRQ 或 nFIQ) 用以唤醒处于空闲模式的 ARM 处理器。通用中断屏蔽特性使能 AIC 来唤醒处理器，不必插入中断，因此在处理器上提供事件同步。

中断源

中断源 0 定义为 FIQ。若产品没有 FIQ 引脚，中断源 0 不能用。

中断源 1 定义为系统中断。它是诸如系统定时器、实时时钟、电源管理控制器及存储控制器等系统外设中断线或的结果。当系统中断出现，服务程序先辨别中断的起因。这可通过读上述系统外设的状态寄存器来实现。

中断源 2 到 31 可与内置的用户外设或外部中断线连接。外部中断线可直接连接或通过 PIO 控制器连接。

PIO 控制器在中断处理中可看作用户外设。因此， PIO 控制器中断线与中断源 2 到 31 连接。

定义在产品上的外设标识符与中断源号码对应 (即位号控制外设时钟)。因此，为简化功能操作及用户接口说明，中断源命名为 FIQ、 SYS 及 PID2 到 PID31。

功能说明

中断源控制

中断源模式

高级中断控制器对每个中断源独立编程。相应的 AIC_SMR(源模式寄存器)SRCTYPE 域中选择每个源的中断条件。

连接在内置外设的输出中断的内部中断源可编程为电平敏感模式或边沿触发模式。内部中断激活电平对用户不重要。

外部中断源可编程为高电平敏感或低电平敏感模式；或正沿触发模式或负沿触发模式。

中断源使能

包括源 0 中的 FIQ 在内的所有中断源均可通过命令寄存器，AIC_IECR (中断使能命令寄存器) 及 AIC_IDCR (中断禁用命令寄存器) , 使能或禁用。该套寄存器通过一条指令控制使能或禁用。可在 AIC_IMR 寄存器中读取中断屏蔽。禁用中断不会影响其它中断服务。

中断清除与设置

所有编程为边沿触发的中断源 (包括源 0 中的 FIQ) 可通过写 AIC_ISCR 与 AIC_ICCR 寄存器来设置与清零。在电平敏感模式下，清除或设置中断源编程无效。

因为在边沿触发模式下对源编程时，软件必须重新初始化记忆电路，因此清除操作不必太认真。但设置操作对于自动测试或软件调试目的有效。它还可用在执行软件中断的 AIC 执行。

当读 AIC_IVR(中断向量寄存器) 时， AIC 自动清除当前中断。只有当该操作影响当前中断时，AIC才会检测该中断源(见See “优先级控制器” on page 136.) 。自动清除降低中断服务程序入口代码读 AIC_IVR 的请求操作。注意，若中断源在快速定向特性使能且仅认为是 FIQ 源时，禁用自动中断清除 (详见 See “ 快速强制 ” on page 138.)。

当读 AIC_FVR 时执行自动清除中断源 0。

中断状态

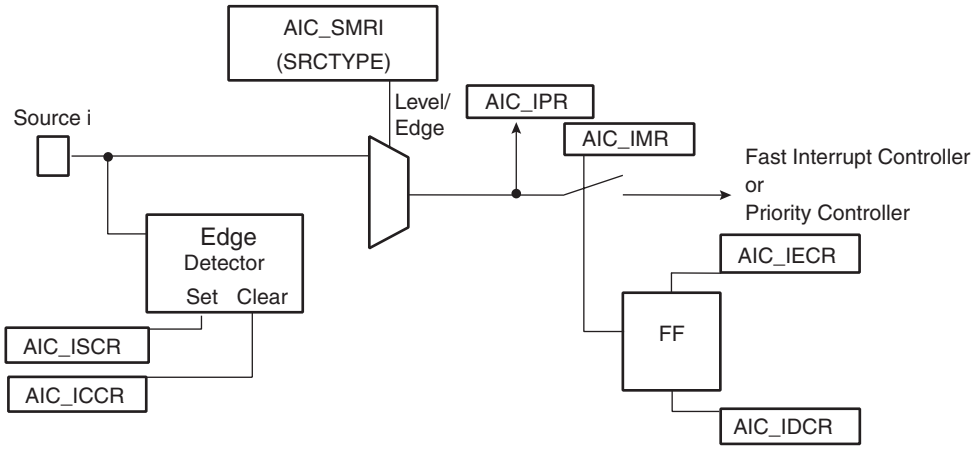
对于每个中断，AIC_IPR(中断挂起寄存器) 与 AIC_IMR (中断屏蔽寄存器) 的屏蔽引起 AIC 操作。 AIC_IPR 使能有效中断源，不管是否被屏蔽。

AIC_ISR寄存器读当前中断数(见“优先级控制器” on page 136) , 寄存器AIC_CISR给出处理器上信号 nIRQ 与 nFIQ 驱动映射。

每个状态可用于优化系统中断处理。

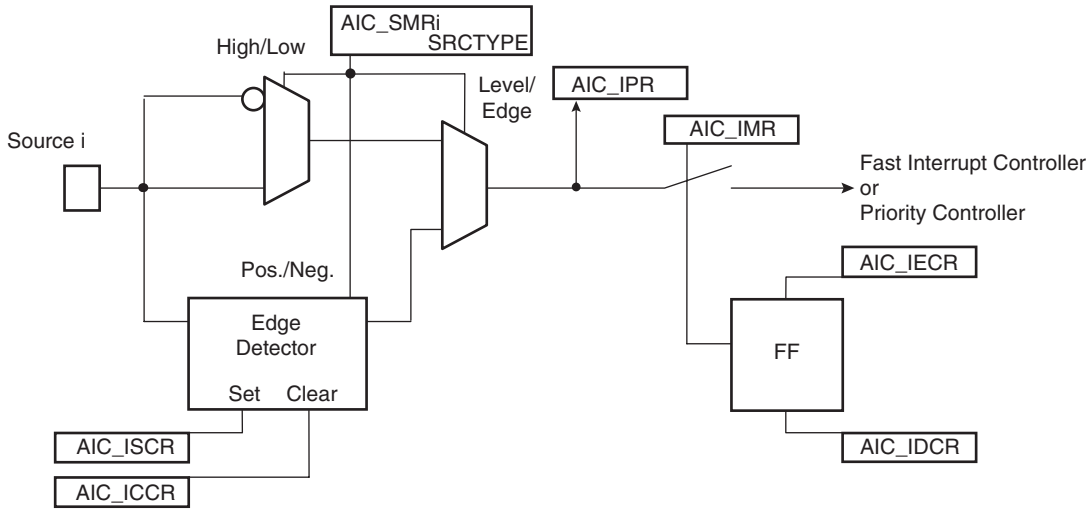
内部中断源输入流程

Figure 46. 内部中断源输入流程



外部中断源输入流程

Figure 47. 外部中断源输入流程



中断延迟

全局中断延迟由以下几个参数决定：

- 软件屏蔽中断时间。
- 处理器级或 AIC 级占用时间。

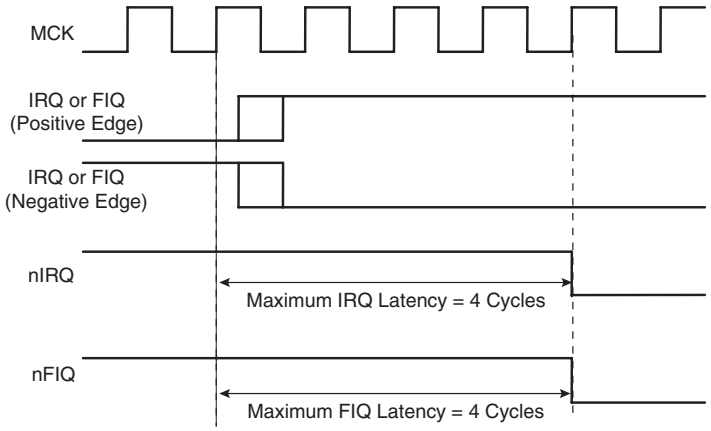
- 当中断出现时处理中的指令执行时间。
- 高优先级中断处理及硬件信号重同步。

该节仅标注了硬件重同步。给出了有效中断 (边沿或电平) 引起的外部中断或内部中断源出现事件到处理器 nIRQ 或 nFIQ 出现间延迟时间的详细说明。重同步时间取决于中断源编程及中断源类型 (内部或外部)。对于标准中断，给出的重同步时间假设没有处理更高优先级的中断。

PIO 控制器复用对于外部中断源的中断延迟没有影响。

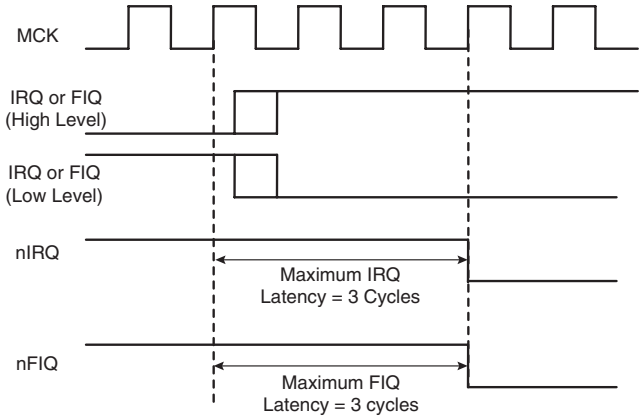
边沿触发外部中断源

Figure 48. 边沿触发外部中断源



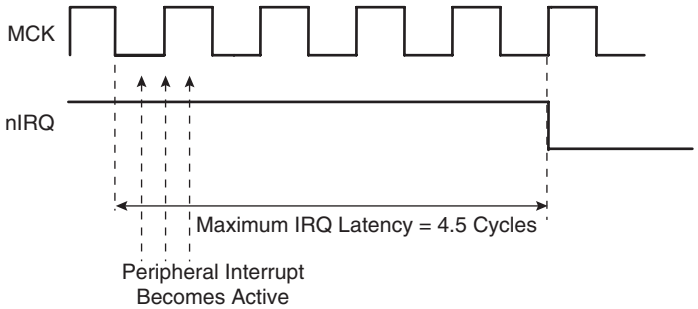
电平敏感外部中断源

Figure 49. 电平敏感外部中断源



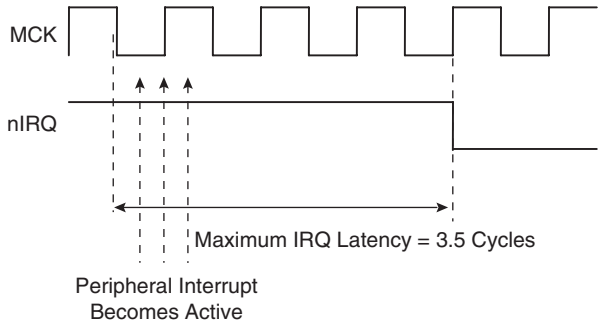
边沿触发内部中断源

Figure 50. 边沿触发内部中断源



电平敏感内部中断源

Figure 51. 电平敏感内部中断源



普通中断

优先级控制器

一个 8 级优先级控制器驱动处理器 nIRQ 线，由出现在中断源 1 到 31 的中断条件决定 (除编程为快速强制的)。

每个中断源通过对相应的 AIC_SMR(源模式寄存器)PRIOR 域写定义其优先级。等级 7 优先级最高，等级 0 优先级最低。

一旦 AIC_SVR (源向量寄存器) 中的 SRCTYPE 域定义的中断条件出现， nIRQ 线出现。由于 nIRQ 出现可能有其它中断源的中断条件出现 ,优先级控制器决定读 AIC_IVR 时的当前中断。对 AIC_IVR 读是中断处理的入口点，允许 AIC 通过软件确定中断。

当前优先级定义为当前中断优先级。

若几个优先级相等的挂起中断源在读 AIC_IVR 后使能，中断源序号最低的中断先行服务。

只有当一个更高优先级的中断源中断条件出现时，nIRQ 线出现。若在中断执行时出现中断条件 (或挂起)，它将延迟到检测到 AIC 结束当前服务。**对 AIC_EOICR 写入将退出中断处理。**

中断嵌套

优先级控制器使用中断嵌套以使得在较低优先级中断服务期间可处理高优先级中断服务。这就需要较低优先级中断服务程序可在处理器级重新使能中断处理。

当处理中断服务期间出现一个中断优先级更高的中断， nIRQ 线重新出现。若中断在内核级使能，打断当前中断执行，新的中断服务应读取 AIC_IVR。此时，当前中断序号及其优先级推入内置硬件堆栈，这样它们得以保存并在高优先级中断服务结束并对 AIC_EOICR 写入后重新加载。

AIC 有 8 级硬件堆栈以便支持有 8 级优先级的中断嵌套。

中断向量

对应于每个中断源的中断处理地址可以保存在寄存器 AIC_SVR1 到 AIC_SVR31 (源向量寄存器 1 到 31) 中。当处理器读 AIC_IVR (中断向量寄存器) ,返回当前中断相关的写入 AIC_SVR 中值。

该特性提供了一种单指令处理当前中断的分支方法，由于 AIC_IVR 映射在绝对地址 0xFFFF F100 处，因而 ARM 中断向量通过下面指令访问地址 0x0000 0018 :

```
LDR   PC,[PC,# -&F20]
```

当处理器执行该指令时，将 AIC_IVR 读出值载入其程序计数器，然后对正确中断程序进行分支处理。

当应用基于操作系统时 (不论是否实时) 不使用该特性。通常操作系统对于所有中断只有一个入口点且其首先执行的是辨别中断源。

但是，强烈推荐 AT91 产品操作系统，其端口支持中断向量。通过定义所有中断源的 AIC_SVR 在其中断句柄地址处由操作系统处理来实现。此时，中断向量允许在特定的快速句柄执行关键中断，而不是操作系统的通用中断处理。这使得对硬件实时任务支持更加有效，且在操作系统中独立运行一个应用程序。

中断处理

本节简单给出使用 AIC 时的快速中断处理序列。假设程序员已了解 ARM 处理器架构，特别是处理器中断模式及相关状态位。

假设如下：

1. 高级中断控制器已编程， AIC_SVR 寄存器载入相应的中断服务程序地址且中断使能。
2. ARM 中断异常向量指令地址须与向量一同作用。

```
LDR PC, [PC, # -&F20]
```

当 nIRQ 出现，若 CPSR 中位 “I” 为 0，序列如下：

1. CPSR 存于 SPSR_irq，程序计数器当前值载入中断链接寄存器 (R14_irq) 且程序计数器 (R15) 中载入 0x18。在由地址 0x1C 取指周期中， ARM 内核调整 R14_irq，以 4 递减。
2. ARM 内核进入中断模式 。
3. 当加载地址为 0x18 的指令执行时，程序计数器载入由 AIC_IVR 读出的值。读 AIC_IVR 有以下效果：
   - 将当前中断挂起并使能最高优先级中断。当前级为当前中断优先级。
   - 将处理器上的 nIRQ 线失效。即使向量未用，必须对 AIC_IVR 读以使 nIRQ 失效。
   - 若编程为边沿触发则自动清除中断。
   - 将当前等级及当前中断序号推入堆栈。
   - 返回相应于当前中断写入 AIC_SVR 的值。
4. 先前的步骤已使跳转到相应的中断服务程序。必须先保存链接寄存器 (R14_irq) 及 SPSR_IRQ。若在中断后直接存于程序计数器，链接寄存器值要以 4 递减。例如，使用指令 `SUB PC, LR, #4` 。
5. 后来的中断可通过清除 CPSR 中的 “I” 位来不加以屏蔽，使得内核又可重新考虑 nIRQ。这在出现了一个高于当前中断优先级的中断时产生。
6. 中断处理程序可按要求执行，开始时保存寄存器，结束时恢复它们的值。在此过程中，高于当前中断优先级的中断将使序列返回步骤 1。

Note: 若中断编程为电平敏感，中断源必须在此过程中清除。

7. 必须将 CPSR 中 “I” 位置位以使在退出前屏蔽中断，保证中断严格按照步骤完成。
8. 须写中断结束命令寄存器 (AIC_EOICR)，以指示 AIC 当前中断已完成。这将使得当前优先级由堆栈中弹出，恢复先前优先级。若有其它中断挂起，等于或低于原先优先级却高于当前新的优先级， nIRQ 线出现出现，但由于 “I” 由内核设置，因此不会立即启动中断序列。恢复 SPSR_irq，最后存于链接寄存器中的值直接写入 PC。这对下面情况有用：中断返回到中断执行前的状态，将存于 SPSR 中的值载入 CPSR，根据保存在 SPSR_irq 中的状态决定是否屏蔽中断。

Note: SPSR 中的 “I” 位非常重要。置位时，则其表示屏蔽指令中断时，ARM 内核处于中断屏蔽的边缘。因此，当 SPSR 恢复时，屏蔽指令完成 (中断被屏蔽)。

快速中断

快速中断源

除使用快速响应特性，中断源 0 是唯一能使处理器发出一个快速中断请求的源。中断源 0 可直接或通过 PIO 控制器与产品的 FIQ 引脚连接。

快速中断控制

AIC 的快速中断逻辑没有优先级控制器。中断源 0 模式在 AIC_SMR0 中编程设置，该寄存器的 PRIOR 域即使读取了要写入的值也不会使用。AIC_SMR0 的 SRCTYPE 域使能编程快速中断源为正边沿或负边沿触发；或高电平或低电平敏感。

对 AIC_IECR (中断使能命令寄存器) 与 AIC_IDCR (中断禁用命令寄存器) 写入 0x1，将分别使能和禁用快速中断。 AIC_IMR (中断屏蔽寄存器) 位 0 指示快速中断是否使能。

快速中断向量

快速中断处理程序地址存于 AIC_SVR0 (源向量寄存器 0)。当处理器读 AIC_FVR (快速向量寄存器) 时将返回写入该寄存器中的值。这提供了一个使用单跳转指令到中断处理的方法，由于

AIC_FVR映射在绝对地址0xFFFF F104 ,通过下面的指令可由地址为0x0000 001C的ARM快速中断向量来访问 :

```
LDR   PC,[PC,# -&F20]
```

当处理器执行该指令时 , 它将由 AIC_FVR 读出的值载入程序计数器 , 然后跳转到快速中断处理程序。若编程为边沿触发模式 , 它还自动执行对快速中断源的清除。

快速中断处理程序

该节给出使用 AIC 时快速中断处理序列。假设程序员已了解 ARM 处理器架构 , 特别是处理器中断模式及相关状态位。

假设如下 :

1. 高级中断控制器已编程 , AIC_SVR0 寄存器载入快速中断服务程序地址且中断源 0 使能。
2. 位于地址 0x1C (FIQ 异常向量地址) 的指令指向快速中断 :

```
LDR PC, [PC, # -&F20]
```

3. 用户不需要嵌套快速中断。

若 CPSR 的位 "F" 为 0 , nFIQ 出现 , 序列为 :

1. CPSR 存于 SPSR_fiq ,程序计数器当前值载入 FIQ 链接寄存器 (R14_FIQ) 且程序计数器 (R15) 中载入 0x1C。在由地址 0x20 取指周期中 , ARM 内核调整 R14_fiq , 以 4 递减。
2. ARM 内核进入 FIQ 模式。
3. 当加载地址为 0x1C 的指令执行时 , 程序计数器载入由 AIC_IVR 读出的值。若编程为边沿触发模式 , 读 AIC_IVR 将自动清除快速中断。仅在此时 , 处理器禁用 nFIQ 线。
4. 先前的步骤已使跳转到相应的中断服务程序。若不需要嵌套快速中断 , 则不用保存链接寄存器 (R14_fiq) 及 SPSR_fiq。
5. 中断处理程序按要求执行。由于 FIQ 自身有专用寄存器且用户 R8 到 R13 为组 , 因此不需要保存寄存器 R8 到 R13。其它寄存器 , R0 到 R7 , 使用前必须保存并在结束后恢复 (在下一步前)。注意 , 若快速中断编程为电平敏感 , 此时必须清除中断源以便释放中断源 0。
6. 最后 , 链接寄存器 R14_fiq 在减 4 后恢复到 PC 中 (例如 , 使用指令 `SUB PC, LR, #4`)。这样将返回中断执行前的状态 , 将 SPSR 中值载入 CPSR 中 , 是否屏蔽快速中断由存于 SPSR 中的状态决定。

Note: SPSR 中的 "F" 位非常重要。如对其置位 , 当屏蔽指令中断时表示 ARM 内核屏蔽 FIQ 中断。因此当 SPSR 恢复时 , 中断指令结束 (FIQ 被屏蔽)。

另一种处理快速中断的方法是将中断服务程序映射到 ARM 向量 0x1C 处。该方法未使用引导向量 , 因此必须在处理程序开始前读 AIC_FVR。但是 , 该方法保存跳转指令的执行。

快速强制

高级中断控制器提供的快速强制提供了任意普通中断源在快速中断控制器改向特性。

快速强制的使能与禁用是通过对快速强制使能寄存器 (AIC_FFER) 及快速强制禁用寄存器 (AIC_FFDR) 的写入来实现。对这些寄存器写操作将更新控制每个内部或外部中断源特性的快速强制状态寄存器 (AIC_FFSR)。

当快速强制禁用后中断源处理如前页所述。

当快速强制使能 , 编程为电平敏感 , 此时中断源的边沿检测仍然有效 , 但源无法触发一个普通中断且不能被优先级处理程序处理。

若中断源编程为电平敏感模式且采得有效电平 , 快速强制结果由 nFIQ 线到内核。

若中断源编程为边沿触发模式且检测到有效边沿 , 快速强制结果由 nFIQ 线到内核。

快速强制特性不影响中断挂起寄存器 (AIC_IPR) 源 0 挂起位。

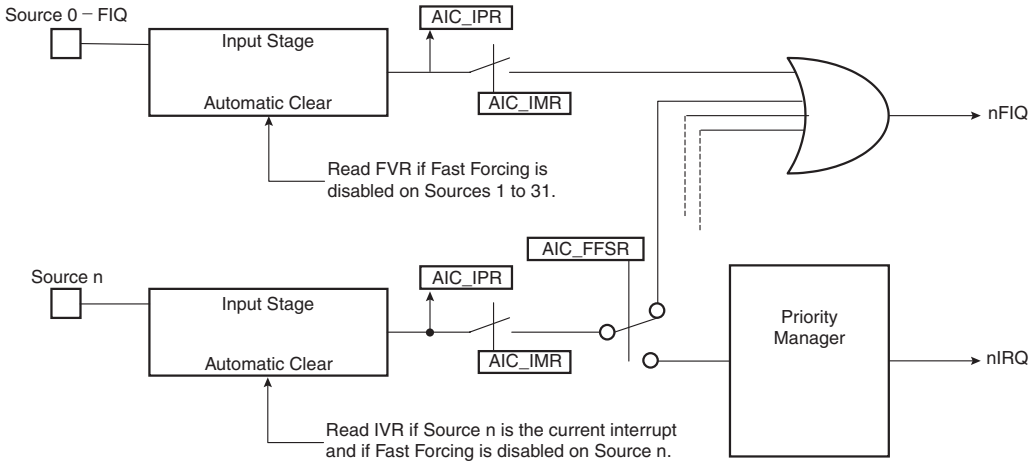
不管是否快速中断源，快速中断向量寄存器 (AIC_FVR) 读源向量寄存器 0 (AIC_SVR0) 中的内容。当使用快速强制特性时，对 FVR 的读取不会将源 0 清除，中断源应通过写中断清除命令寄存器 (AIC_ICCR) 来清除。

所有有快速强制特性使能且编程为边沿触发模式的使能或挂起的中断源必须通过写中断清除命令寄存器来清除。这样，它们将独立清除且防止丢失中断。

读 AIC_IVR 不会清除已使能了快速强制特性的中断源。

保留给快速中断的源 0 维持正常操作，成为一个快速中断源。

Figure 52. 快速强制



保护模式

保护模式允许在不执行相关自动操作时读中断向量寄存器。这对工作在调试系统下的工作来说是必要的。当调试器与调试监控器或 ARM 处理器的 ICE 一起工作时，停止应用程序并更新打开的窗口，可能会读 AIC 用户接口及 IVR。下面是一些不希望见到的结果：

- 若有一个高于当前中断优先级的中断挂起，将其推入堆栈。
- 若没有使能挂起中断，返回先前向量。

上述两种情况下中断结束命令需要得到应答且恢复 AIC 的前后关系。通常该操作不是由调试系统执行，因为调试系统将成为强烈的干扰并引起应用程序进入非理想状态。

这可通过使用保护模式来避免。在 AIC_DCR (调试控制寄存器) 的 DBGM 写 0x1 使能保护模式。

保护模式下，只有当对 AIC_IVR 执行写访问时， AIC 执行中断入栈。因此，中断服务程序在读 AIC_IVR 后必须写入判决数据。AIC 新的前后关系，包括中断状态寄存器 (AIC_ISR) 值，只有在 AIC_IVR 写入后方能更新当前中断。

AIC_IVR读本身的值(如，通过调试器)，不会改变AIC前后关系及AIC_ISR内容。额外的AIC_IVR读执行相同的操作。但是，建议在中断服务程序 AIC_IVR 读写间不要停止处理器运行，以保证调试器不修改 AIC 前后关系。

正常操作模式下，在 AIC 中对 AIC_IVR 的读取执行了下列操作：

1. 计算有效中断 (是否高于当前优先级或是假的)。
2. 确定并返回有效中断向量。
3. 存储中断。
4. 将当前优先级推入内部堆栈。
5. 确认中断。

但在保护模式中，读 AIC_IVR 时只需执行 1 到 3 步 ；当写 AIC_IVR 时执行 4 与 5 步。

保护模式下写与调试程序可在正常模式下正确运行而不用加以修改。但是，普通模式下，AIC_IVR 写无影响，因此可将其删除以优化代码。

伪中断

高级中断控制器特性防止伪中断。伪中断的定义是中断源长足以在 AIC 上插入 nIRQ，但当 AIC_IVR 读时已消失。最有可能出现在：

- 外部中断源编程在电平敏感模式下且有效电平仅出现极短的时间。
- 外部中断源编程在电平敏感模式下且内置外设相应的输出信号仅激活极短的时间。
- 软件屏蔽前中断只出现几个周期，因此导致中断源出现脉冲。

AIC 在 AIC_IVR 读且中断源挂起未使能时检测到伪中断。此时，AIC 返回值存于 AIC_SPU (伪向量寄存器) 中。程序员在 AIC_SPU 中保存伪中断处理程序地址作为应用程序的一部分，以尽快返回正常执行流程。写入 AIC_EOICR 的处理程序执行中断返回。

通用中断屏蔽

AIC 的通用中断屏蔽位防止中断传递到处理器。若 AIC_DCR (调试控制寄存器) 中的 GMSK 位置位，nIRQ 与 nFIQ 线将驱动到它们的停止状态。但是该屏蔽不妨碍处理器由空闲模式下的唤醒。该功能方便处理器在下一事件上同步，当事件出现后不用处理中断，立即执行并发操作。建议小心使用该屏蔽。

高级中断控制器 (AIC) 用户接口

基地址

AIC映射到地址**0xFFFF F000**上。共有4-K字节的地址空间。因ARM处理器与PC相关的下载/保存指令仅支持 ± 4-K 字节的偏移，因此允许向量引导特性。

Table 46. 高级中断控制器 (AIC) 寄存器映射

偏移	寄存器	名称	访问类型	复位值
0000	源模式寄存器 0	AIC_SMR0	读 / 写	0x0
0x04	源模式寄存器 1	AIC_SMR1	读 / 写	0x0
–	–	–	–	–
0x7C	源模式寄存器 31	AIC_SMR31	读 / 写	0x0
0x80	源向量寄存器 0	AIC_SVR0	读 / 写	0x0
0x84	源向量寄存器 1	AIC_SVR1	读 / 写	0x0
–	–	–	–	–
0xFC	源向量寄存器 31	AIC_SVR31	读 / 写	0x0
0x100	中断向量寄存器	AIC_IVR	只读	0x0
0x104	快速中断向量寄存器	AIC_FVR	只读	0x0
0x108	中断状态寄存器	AIC_ISR	只读	0x0
0x10C	中断挂起寄存器	AIC_IPR	只读	0x0[1]
0x110	中断屏蔽寄存器	AIC_IMR	只读	0x0
0x114	内核中断状态寄存器	AIC_CISR	只读	0x0
0x118	保留	–	–	–
0x11C	保留	–	–	–
0x120	中断使能命令寄存器	AIC_IECR	只写	–
0x124	中断禁用命令寄存器	AIC_IDCR	只写	–
0x128	中断清除命令寄存器	AIC_ICCR	只写	–
0x12C	中断置位命令寄存器	AIC_ISCR	只写	–
0x130	中断结束命令寄存器	AIC_EOICR	只写	–
0x134	伪中断向量寄存器	AIC_SPU	读 / 写	0x0
0x138	调试控制寄存器	AIC_DCR	读 / 写	0x0
0x13C	保留	–	–	–
0x140	快速强制使能寄存器	AIC_FFER	只写	–
0x144	快速强制禁用寄存器	AIC_FFDR	只写	–
0x148	快速强制状态寄存器	AIC_FFSR	只读	0x0

Note: 1. 中断挂起寄存器复位值由外部中断源电平决定。其它所有中断源在复位时清零而不挂起。

AIC 源模式寄存器

寄存器名称：AIC_SMR0..AIC_SMR31

访问类型： 读 / 写

复位值： 0x0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	SRCTYPE		–	–	PRIOR		

- **PRIOR: 优先级**

对除 FIQ 源 (源 0) 外的中断源优先级编程。

优先级在 0 (最低) 到 7 (最高) 之间。

优先级没有在与 SMR 寄存器 AIC_SMRx 相关的 FIQ 中使用。

- **SRCTYPE: 中断源类型**

内部中断源有效电平或边沿不可编程。

SRCTYPE		内部中断源
0	0	电平敏感
0	1	边沿触发
1	0	电平敏感
1	1	边沿触发

AIC 源向量寄存器

寄存器名称： AIC_SVR0..AIC_SVR31

访问类型： 读 / 写

复位值： 0x0

31	30	29	28	27	26	25	24
VECTOR							

23	22	21	20	19	18	17	16
VECTOR							

15	14	13	12	11	10	9	8
VECTOR							

7	6	5	4	3	2	1	0
VECTOR							

- **VECTOR: 源向量**

用户可在这些寄存器中存储各个中断源相关处理程序地址。

AIC 中断向量寄存器

寄存器名称： AIC_IVR

访问类型： 只读

复位值： 0

31	30	29	28	27	26	25	24
IRQV							

23	22	21	20	19	18	17	16
IRQV							

15	14	13	12	11	10	9	8
IRQV							

7	6	5	4	3	2	1	0
IRQV							

- **IRQV: 中断向量寄存器**

中断向量寄存器包含了用户编程的当前中断相关的源向量寄存器。

读中断向量寄存器时，源向量寄存器使用中断号码作为索引。

当没有当前中断，中断向量寄存器读取存储于 AIC_SPU 中的值。

AIC FIQ 向量寄存器

寄存器名称： AIC_FVR

访问类型： 只读

复位值： 0

31	30	29	28	27	26	25	24
FIQV							
23	22	21	20	19	18	17	16
FIQV							
15	14	13	12	11	10	9	8
FIQV							
7	6	5	4	3	2	1	0
FIQV							

- **FIQV: FIQ 向量寄存器**

FIQ向量寄存器包含用户在源向量寄存器0中编程的向量值。当无快速中断时，快速中断向量寄存器读存于AIC_SPU中的值。

AIC 中断状态寄存器

寄存器名称： AIC_ISR

访问类型： 只读

复位值： 0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	IRQID				

- **IRQID: 当前中断标识**

中断状态寄存器返回当前中断源序号。

AIC 中断挂起寄存器

寄存器名称： AIC_IPR

访问类型： 只读

复位值： 0

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: 中断挂起**

0 = 相关中断未挂起。

1 = 相关中断挂起。

AIC 中断屏蔽寄存器

寄存器名称： AIC_IMR

访问类型： 只读

复位值： 0

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: 中断屏蔽**

0 = 相应中断禁用。

1 = 相应中断使能。

AIC 内核中断状态寄存器

寄存器名称： AIC_CISR

访问类型： 只读

复位值： 0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	NIRQ	NIFQ

- **NFIQ: NFIQ 状态**

0 = nFIQ 线无效。

1 = nFIQ 线激活。

- **NIRQ: NIRQ 状态**

0 = nIRQ 线无效。

1 = nIRQ 线激活。

AIC 中断使能命令寄存器

寄存器名称： AIC_IECR

访问类型： 只写

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID3: 中断使能**

0 = 无效。

1 = 使能相应中断。

AIC 中断禁用命令寄存器

寄存器名称： AIC_IDCR

访问类型： 只写

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: 中断禁用**

0 = 无效。

1 = 禁用相应中断。

AIC 中断清除命令寄存器

寄存器名称： AIC_ICCR

访问类型： 只写

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: 中断清除**

0 = 无效。

1 = 清除相应中断。

AIC 中断置位命令寄存器

寄存器名称：AIC_ISCR

访问类型： 只写

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	FIQ

- **FIQ, SYS, PID2-PID31: 中断置位**

0 = 无效。

1 = 置位相应中断。

AIC 中断结束命令寄存器

寄存器名称： AIC_EOICR

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

中断处理程序用中断结束命令寄存器指示中断处理结束。因为只需要向该寄存器地址写值以标识中断处理结束，所以可向其写入任意值。

AIC 伪中断向量寄存器

寄存器名称： AIC_SPU

访问类型：读 / 写

复位值： 0

31	30	29	28	27	26	25	24
SIQV							
23	22	21	20	19	18	17	16
SIQV							
15	14	13	12	11	10	9	8
SIQV							
7	6	5	4	3	2	1	0
SIQV							

- **SIQV: 伪中断向量寄存器**

用户可在该寄存器中存储伪中断处理程序的地址。当出现未中断时，写入该寄存器的值为 AIC_IVR 的返回值，类似的发生伪快速中断时，写入该寄存器的值为 AIC_IVR 的返回值。

AIC 调试控制寄存器

寄存器名称： AIC_DEBUG

访问类型 :读 / 写

复位值 :0

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	GMSK	PROT

- **PROT: 保护模式**

0 = 保护模式禁用。

1 = 保护模式使能。

- **GMSK: 产生屏蔽**

0 = nIRQ 与 nFIQ 线由 AIC 控制。

1 = nIRQ 与 nFIQ 线置于无效状态。

AIC 快速强制使能寄存器

寄存器名称： AIC_FFER

访问类型 :只写

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	–

- **SYS, PID2-PID31: 快速强制使能**

0 = 无效。

1 = 使能相应中断快速强制特性。

AIC 快速强制禁用寄存器

寄存器名称： AIC_FFDR

访问类型 :只写

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	–

- **SYS, PID2-PID31: 快速强制禁用**

0 = 无效。

1 = 禁用相应中断快速强制特性。

AIC 快速强制状态寄存器

寄存器名称：AIC_FFSR

访问类型 :只读

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	SYS	–

- **SYS, PID2-PID31: 快速强制状态**

0 = 禁用相应中断的快速强制特性。

1 = 使能相应中断的快速强制特性。

时钟发生器

说明

时钟发生器由 1 个 PLL，1 个主振荡器，一个 RC 振荡器组成，它提供下列时钟：

- SLCK，慢速时钟，系统内唯一恒定时钟。
- MAINCK 为主振荡器输出。
- PLLCK 为分频器与 PLL 块输出。

时钟发生器用户接口内置在电源管理控制器中，说明见 “ 电源管理控制器 (PMC) 用户接口 ” on page 164。时钟发生器寄存器名称为 CKGR_。

慢速时钟 RC 振荡器

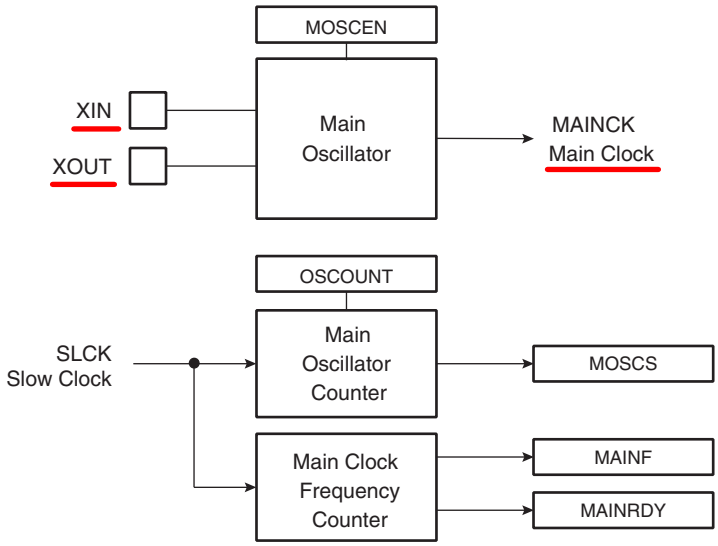
慢速时钟是 RC 振荡器输出，且是包括电源管理控制器在内的系统内唯一恒定时钟。它是 PMC 强制操作。

用户必须将可能的 RC 振荡器漂移计算在内，详见 DC 特性部分。

主振荡器

Figure 53 给出主振荡器方框图。

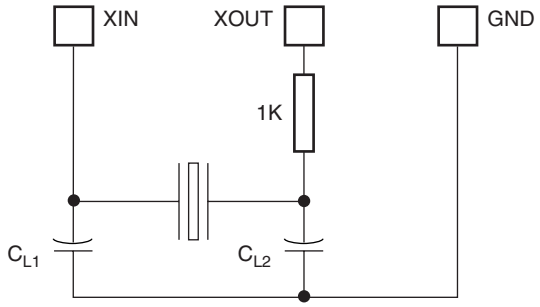
Figure 53. 主振荡器方框图



主振荡器连接

时钟发生器中集成的主振荡器为 3 ~ 20 MHz 的基频晶体。典型的晶体连接见 Figure 54。只有在频率低于 8 MHz 时才需要 1 kΩ 电阻。振荡器每个 XIN 与 XOUT 引脚 上有一个 25 pF 的电容。因此，当晶体有 12.5 pF 的负载电容时，将 CL1 与 CL2 去掉。更多关于主振荡器的电气特性，见 DC 特性部分。

Figure 54. 典型晶体连接



主振荡器启动时间

主振荡器启动时间见 DC 特性部分。启动时间由晶体频率决定，频率上升时间减少。

主振荡器控制

为优化系统启动的电源需求，复位后主振荡器禁用并使用慢速时钟。

通过清除主振荡器寄存器 (CKGR_MOR) 的 MOSCEN 位来禁用主振荡器以减低功耗。

当通过清除 CKGR_MOR 中的 MOSCEN 位将主振荡器禁用时， PMC_SR 中的 MOSCS 位自动清除，即表示主时钟关闭。

当使能主振荡器时，用户必须用对应于振荡器启动时间的值初始化主振荡器计数器。启动时间由与主振荡器连接的晶体频率确定。

当 MOSCEN 位及 OSCOUNT 写入 CKGR_MOR 后使能主振荡器， PMC_SR(状态寄存器) 中的 MOSCS 位清零且计数器按照慢时钟 8 分频的速度向下对 OSCOUNT 值开始计数。由于 OSCOUNT 值为 8 位，因此最大启动时间为 62 ms。

当计数器达到 0， MOSCS 位置位表示主时钟有效。设置 PMC_IMR 中的 MOSCS 位可触发处理器中断。

主时钟频率计数器

主振荡器的主时钟频率计数器提供与主振荡器连接的石英频率。通常，该值由系统设计者确定；但是它对引导程序正确配置时钟速度非常有用。

当主振荡器稳定后，即 MOSCS 置位后，在慢时钟下一个上升沿时，主时钟频率计数器以主时钟速度开始向上计数。然后在慢时钟第 16 个下降沿时，CKGR_MCFR(主时钟频率寄存器) 中的 MAINRDY 位置位且计数器停止计数。其值可从 CKGR_MCFR 中的 MAINF 域读出且给出 16 个慢时钟中主时钟周期数，这样可得到与主振荡器连接的晶体频率。

主振荡器旁路

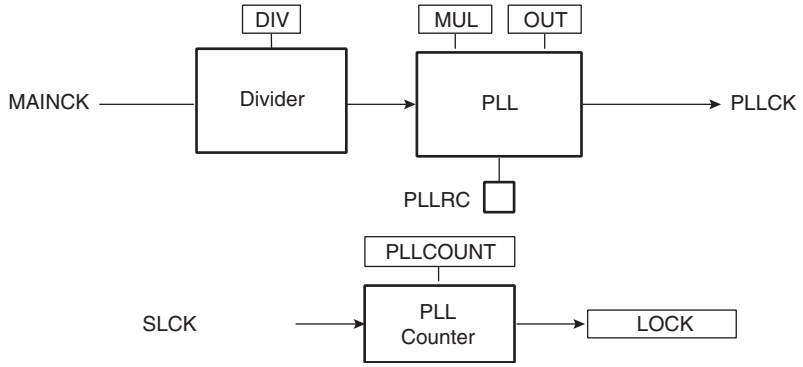
用户向器件提供时钟可不通过连接晶体。此时，用户必须在 XIN 引脚提供外部时钟信号。此时 XIN 引脚输入的符号见产品电气特性部分。程序员必须将主 OSC 寄存器 (CKGR_MOR) 的 OSCBYPASS 位设置为 1 而 MOSCEN 位设置为 0，以保证外部时钟正常工作。

分频器与 PLL 模块

PLL 内置一个输入分频器以增加结果时钟信号的精度。但是，当对分频器编程时，用户必须考虑 PLL 最小输入频率。

Figure 55 给出分频器与 PLL 模块方框图。

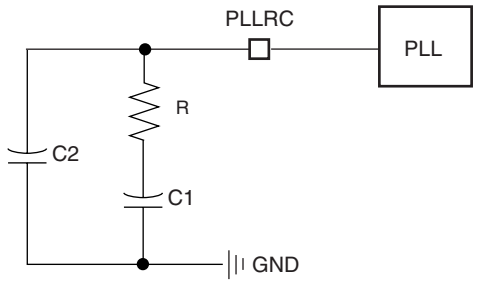
Figure 55. 分频器与 PLL 模块方框图



PLL 滤波器

PLL 需要通过 PLLRC 引脚与一个外部二阶滤波器连接。 Figure 56 给出滤波器示意图。

Figure 56. PLL 电容与电阻



与 PLLRC 连接的 R、C1 与 C2 值由 PLL 输入频率、PLL 输出频率及相位容限确定。必须在输出信号过量与启动时间间找到平衡。

分频器与锁相环编程

分频器可在 1 到 255 间置位，步长为 1。当分频器域 (DIV) 设置为 0，相应分频器及 PLL 输出为连续的 0 信号。复位时，每个 DIV 域置为 0，因此相应的 PLL 输入时钟置为 0。

PLL 允许分频器输出相乘。PLL 时钟信号频率由各自源信号频率及参数 DIV 与 MUL 确定。源信号频率系数为 (MUL + 1)/DIV。当 MUL 写入 0，相应的 PLL 禁用并节省其功耗。在 MUL 域写入大于 0 的值将重新使能 PLL。

当 PLL 重新使能或它的某个参数改变， PMC_SR 中的 LOCK 位自动清零。 CKGR_PLLR 中 PLLCOUNT 域值载入 PLL 计数器。PLL 计数器开始以慢时钟速率开始递减直到其值为 0。此时，LOCK 位置位并能触发处理器中断。用户须在 PLLCOUNT 域载入所需慢时钟周期数来覆盖 PLL 过渡时间。过渡时间由 PLL 滤波器确定。PLL 初始状态及其目标频率可使用 Atmel 提供的专用工具进行计算。

电源管理控制器 (PMC)

说明

电源管理控制器 (PMC) 通过控制系统及用户外设时钟来优化功耗。PMC 使能 / 禁用许多外设及 ARM 处理器的时钟输入。

电源管理控制器提供下列时钟：

- MCK，主机时钟，可编程，其频率由几百 Hz 到器件最高工作频率。可用在恒定运行的模块中，如 AIC 或存储控制器。
- 处理器时钟 (PCK)，当处理器进入空闲模式时关闭。

- 外设时钟，代表为 MCK，提供给内置外设 (USART、SSC、SPI、TWI、TC、MCI 等) 并可独立控制。为减少产品中时钟名称数目，产品手册中将外设时钟称为 MCK。
- UDP 时钟 (UDPCK)， USB 器件端口工作时需要。
- 可编程时钟输出可从时钟发生器提供的时钟中选择并在 PCKx 引脚上驱动。

主机时钟控制器

主机时钟控制器提供主机时钟 (MCK) 的选择与分频。 MCK 为所有外设及存储控制器时钟。

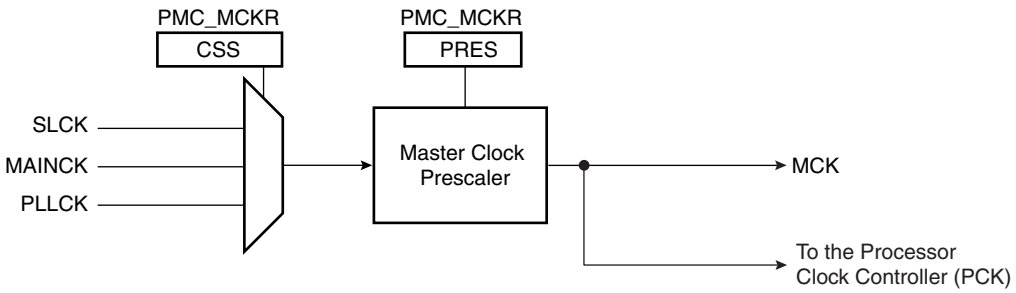
主机时钟由时钟发生器提供的时钟中选择。选择慢时钟则向整个器件提供一个慢时钟信号。选择主时钟会节省 PLL 功耗。

主机时钟控制器由时钟选择器及预分频器组成。

通过写 PMC_MCKR (主机时钟寄存器) 的 CSS 域 (时钟源选择) 对主机时钟进行选择。预分频器分频因子为在 1 到 64 间 2 的幂次。 PMC_MCKR 中的 PRES 域对预分频器编程。

每次 PMC_MCKR 写入定义一个新的主机时钟，PMC_SR 中的 MCKRDY 位清除。在主机时钟建立前它的值为 0。然后，MCKRDY 位被置位并能触发处理器中断。该特性在当由高速时钟向低速时钟切换，用来通知何时改变。

Figure 57. 主机时钟控制器



处理器时钟控制器

PMC 是使处理器时钟控制器 (PCK) 执行处理器空闲模式。通过写系统时钟使能寄存器 (PMC_SCER) 及系统时钟禁用寄存器 (PMC_SCDR) 可使能或禁用处理器时钟。该时钟状态可在系统时钟状态寄存器 (PMC_SCSR) 中读取。

处理器时钟 PCK 在复位后使能并可通过任意使能中断自动重新使能。通过禁用处理器时钟进入处理器空闲模式，而处理器时钟可由任意使能的快速或普通中断重新使能，或由产品复位来使能。

当处理器时钟禁用，当前指令在时钟停止前结束，但这不能防止数据由其它主机通过系统总线的传输。

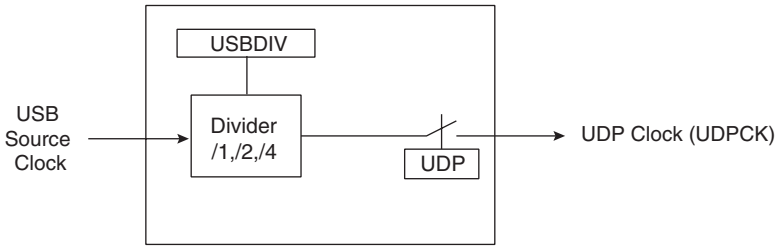
USB 时钟控制器

USB源时钟为PLL输出。若使用USB，用户必须使用CKGR_PLLR中的USBDIV位编程使PLL产生精度在 ± 0.25% 的 48 MHz、 96 MHz 或 192 MHz 信号 。

当 PLL 输出稳定，即 LOCK 位置位：

- 通过设置 PMC_SCER 中的 UDP 位将使能 USB 器件时钟。当该外设不使用时，可设置 PMC_SCDR中的UDP位以节省功耗。UDP位可激活该时钟。USB器件端口需要48 MHz信号及主机时钟。外设时钟控制器控制主机时钟。

Figure 58. USB 时钟控制器



外设时钟控制器

电源管理控制器通过外设时钟控制器来控制每个内置外设时钟。用户可通过对外设时钟使能(PMC_PCER)及外设时钟禁用(PMC_PCDR)寄存器来独立使能或禁用外设的主机时钟。在外设时钟状态寄存器 (PMC_PCSR) 中可读取外设时钟状态。

当外设时钟禁用时，该时钟立即停止。外设时钟在复位后自动禁用。

停止外设时，建议在禁用时钟前执行完最后一条指令。以避免数据出错或系统出错。

外设时钟控制寄存器位序号 (PMC_PCER、PMC_PCDR 及 PMC_PCSR) 为定义在产品级的外设标识符。通常，该序号对应于分配给外设的中断源序号。

可编程时钟输出控制器

PMC 控制外部引脚 PCKx 的 3 个输出信号。每个信号均可通过 PMC_PCKx 寄存器独立编程。

通过写 PMC_PCKx 中的 CSS 域，PCKx 可在慢时钟、PLL 输出及主时钟间选择。每个输出信号可由 1 到 64 间 2 的幂次进行分频，具体系数在 PMC_PCKx 中的 PRES (预分频) 域写入。

通过在 PCKx 中 PMC_SCER 或 PMC_SCDR 位写入 1 来使能或禁用输出信号。工作的可编程时钟状态由 PMC_SCSR(系统时钟状态寄存器) 中的 PCKx 位给出。

此外，与 PCK 类似，PMC_SR 中的状态位表示可编程时钟实际上是对可编程时钟寄存器的什么编程。

由于可编程时钟控制器不会管理当切换时钟时出现的脉冲，强烈建议在配置变化前将可编程时钟禁用并在变化后重新使能。

编程序列

1. 使能主振荡器：

   通过设置 CKGR_MOR 寄存器中的 MOSCEN 域来使能主振荡器。某些情况下定义一个启动时间是有利的。可通过在 CKGR_MOR 寄存器中的 OSCOUNT 域写入值实现。

   一旦该寄存器正确配置，用户必须等待 PMC_SR 寄存器中的 MOSCS 域置位。可通过轮询状态寄存器或等待当 PMC_IER 寄存器中与 MOSCS 相关的中断的中断线上升。

   代码示例：

   ```
   write_register(CKGR_MOR,0x00000701)
   ```

   启动时间 = 8 * OSCOUNT / SLCK = 56 慢时钟周期。

   因此，在 56 慢时钟周期后，主振荡器将使能 (MOSCS 位置位)。

2. 校验主振荡器频率 (可选)：

   某些情况下用户需要精确测量主振荡器频率。测量通过 CKGR_MCFR 寄存器来完成。

   一旦CKGR_MCFR寄存器中的MAINRDY域置位，用户可读CKGR_MCFR寄存器的MAINF域。它提供 16 个慢时钟周期中主时钟周期数。

3. 设置 PLL 与分频器：

   PLL 与分频器所需配置的参数均位于 CKGR_PLLR 寄存器。

   DIV域用来控制分频器本身。可为0到255间值。分频器输出是其输入除以DIV参数的结果。DIV 参数默认值为 0，表示分频器关闭。

OUT 域用来选择 PLL B 输出频率范围。

MUL 域为 PLL 乘数。该参数可为 0 到 2047 间值。若 MUL 置为 0，PLL 将关闭，否则 PLL 输出频率为 PLL 输入频率与 (MUL + 1) 的乘积。

PLLCOUNT 域给出在 CKGR_PLLR 写入后 PMC_SR 寄存器 LOCK 位置位前慢时钟周期数。

一旦 PMC_PLL 寄存器写入，用户必须等待 PMC_SR 寄存器的 LOCK 位置位。这可通过轮询状态寄存器或等待 PMC_IER 寄存器中与 LOCK 相关的中断使能时的中断线上升来实现。CKGR_PLLR 中所有参数可在单写操作中编程。若下列参数处于某一阶段，MUL、DIV 修改，LOCK 位变低以表示 PLL 未就绪。当 PLL 锁定，LOCK 重新设置。在使用 PLL 输出时钟前，强迫用户等待 LOCK 位置位。

USBDIV 域用来控制附加分频器，1、2 或 4，用来产生 USB 时钟。

代码示例：

```
write_register(CKGR_PLLR,0x00040805)
```

若 PLL 与分频器使能，PLL 输入时钟为主时钟。PLL 输出时钟为 PLL 输入时钟的 5 倍。若 CKGR_PLLR 写入，LOCK 位在 8 个慢时钟周期后置位。

4. 选择主机时钟与处理器时钟

主机时钟与处理器时钟是通过 PMC_MCKR 寄存器配置的。

CSS 域用来选择主机时钟分频器源。默认的源为慢时钟。

PRES 域用来控制主机时钟预分频器。用户可选择不同的值 (1、2、4、8、16、32、64)。主机输出为预分频输入被 PRES 分频后的值。PRES 默认值为 1，表示主机时钟等于慢时钟。

一旦 PMC_MCKR 寄存器写入，用户必须等待 PMC_SR 寄存器中的 MCKRDY 位置位。这可通过对状态寄存器轮询或等待 PMC_IER 寄存器中与 MCKRDY 相关的中断使能的中断线上升。

PMC_MCKR 中所有参数可在单写操作中完成。若 CSS 或 PRES 修改，MCKRDY 位将变低表示主机时钟与处理器时钟未就绪。用户在使用主机与处理器时钟前必须等到 MCKRDY 位重新置位。

Note: 若选择 PLLx 时钟作为主机时钟且用户决定通过写 CKGR_PLLR 来对其进行修改，MCKRDY 标志将变低而 PLL 解锁。一旦 PLL 重新上锁，LOCK 变高且 MCKRDY 置位。
若 PLL 解锁，主机时钟自动选择变为主时钟，详见 “ 时钟切换波形 ” on page 162。

代码示例：

```
write_register(PMC_MCKR,0x00000011)
```

主机时钟为主时钟 16 分频。

处理器时钟为主机时钟。

5. 可编程时钟选择

可编程时钟通过 PMC_SCER、PMC_SCDR 及 PMC_SCSR 寄存器进行控制。

可编程时钟可通过 PMC_SCER 或 PMC_SCDR 寄存器来使能或禁用。根据系统需求，对 3 个可编程时钟进行使能或禁用。PMC_SCSR 显示哪个可编程时钟使能。默认情况下所有可编程时钟全部禁用。

PMC_PCKx 寄存器用来配置可编程时钟。

CSS 域用来选择可编程时钟分频源。有四个可选时钟：主时钟，慢时钟 PLLCK。默认情况下选择慢时钟。

PRES 域用来控制可编程时钟预分频器。它可在不同值间进行选择 (1、2、4、8、16、32、64)。可编程时钟输出为预分频输入被 PRES 分频后的值。PRES 默认值为 1 ,表示主机时钟等于慢时钟。

一旦 PMC_PCKx 寄存器编程，相应的可编程时钟必须使能且用户必须等待 PMC_SR 寄存器中的 PCKRDYx 位置位。这可通过对状态寄存器轮询或等待 PMC_IER 寄存器中与 PCKRDYx 相关的中断使能的中断线上升。PMC_PCKx 中所有参数编程可在单写操作中完成。

若 CSS 与 PRES 参数被修改，必须先禁用相应的可编程时钟。然后才能修改参数。一旦将可编程时钟禁用，用户必须重新使能可编程时钟并等待 PCKRDYx 位置位。

代码示例：

```
write_register(PMC_PCK0,0x00000015)
```

可编程时钟 0 为主时钟 32 分频。

6. 使能外设时钟

一旦完成所有前面的步骤，可通过 PMC_PCER 或 PMC_PCDR 寄存器对外设时钟进行使能或禁用。

根据系统需求对 9 个外设时钟进行使能或禁用。 PMC_PCSR 显示哪个外设时钟使能。

Note: 每个使能的外设时钟对应主机时钟。

代码示例：

```
write_register(PMC_PCER,0x00000110)
```

外设时钟 4 与 8 使能。

```
write_register(PMC_PCDR,0x00000010)
```

外设时钟 4 禁用。

时钟切换

主机时钟切换时间

Table 47 给出主机时钟切换最差情况下的时间需求。此时预分频器未激活。当预分频器激活 ,增加新选择时钟的 64 个时钟周期。

Table 47. 时钟切换时间 (最差情况)

从 到	主时钟	SLCK	PLL 时钟
主时钟	–	4 x SLCK + 2.5 x 主时钟	3 x PLL 时钟 + 4 x SLCK + 1 x 主时钟
SLCK	0.5 x 主时钟 + 4.5 x SLCK	–	3 x PLL 时钟 + 5 x SLCK
PLL 时钟	0.5 x 主时钟 + 4 x SLCK + PLLCOUNT x SLCK + 2.5 x PLLx 时钟	2.5 x PLL 时钟 + 5 x SLCK + PLLCOUNT x SLCK	2.5 x PLL 时钟 + 4 x SLCK + PLLCOUNT x SLCK

时钟切换波形

Figure 59. 主机时钟由慢时钟切换到 PLL 时钟

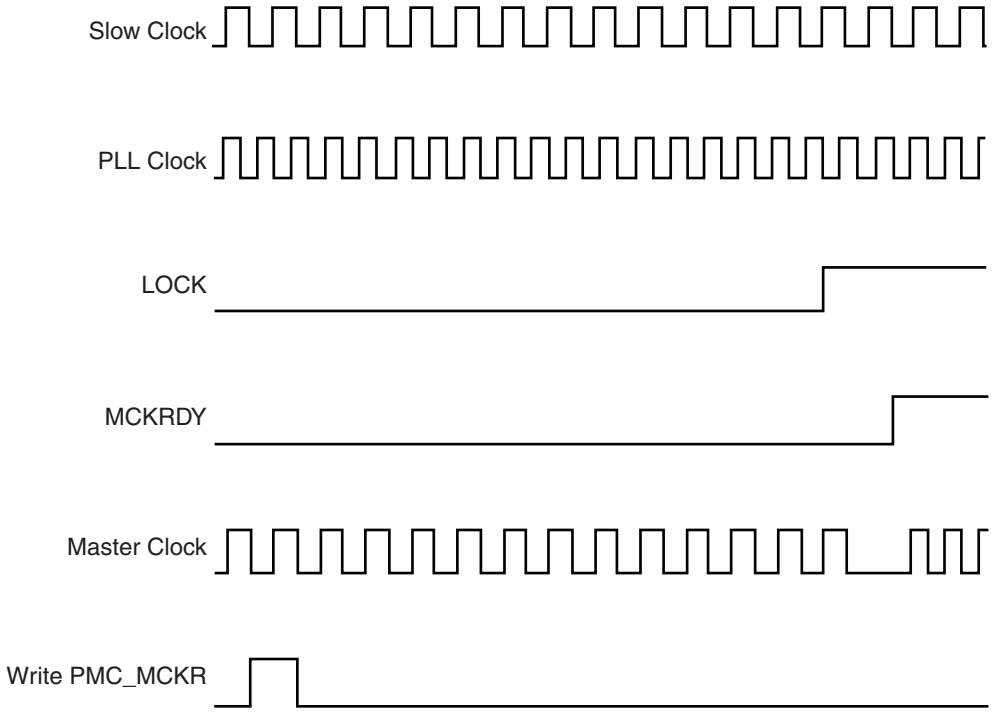


Figure 60. 主机时钟由主时钟切换到慢时钟

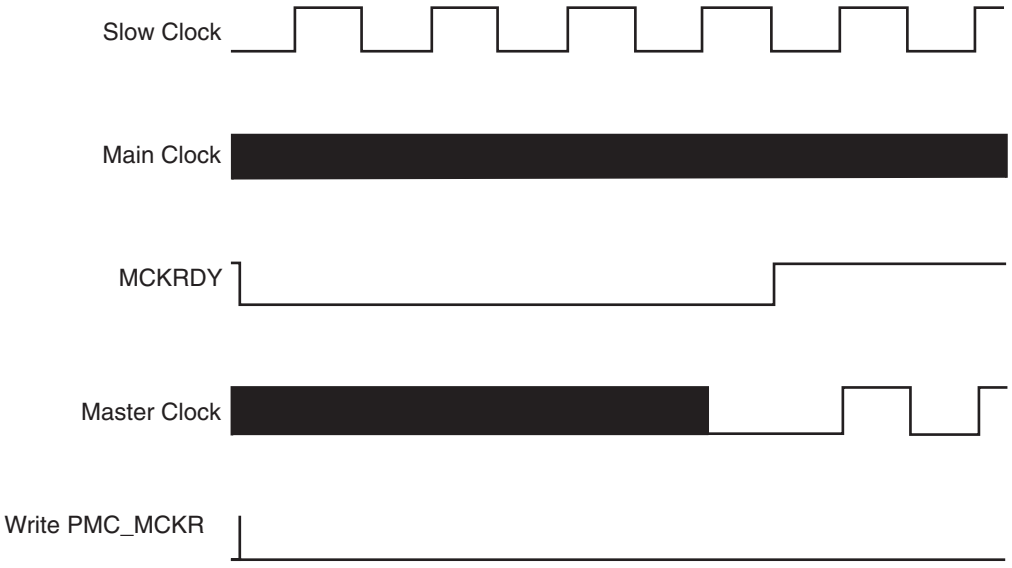


Figure 61. 改变 PLL 编程

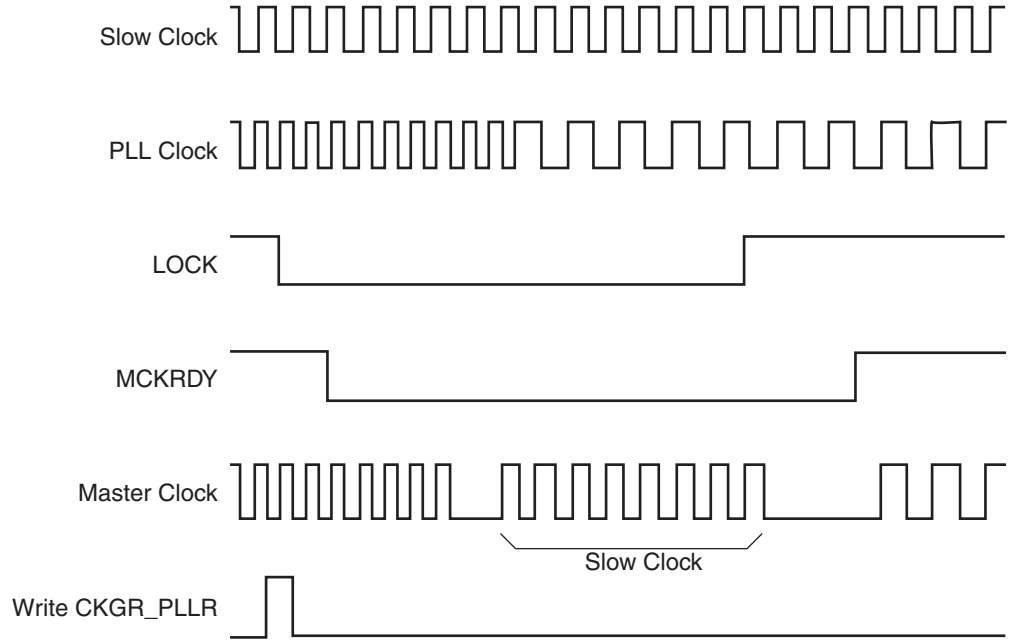
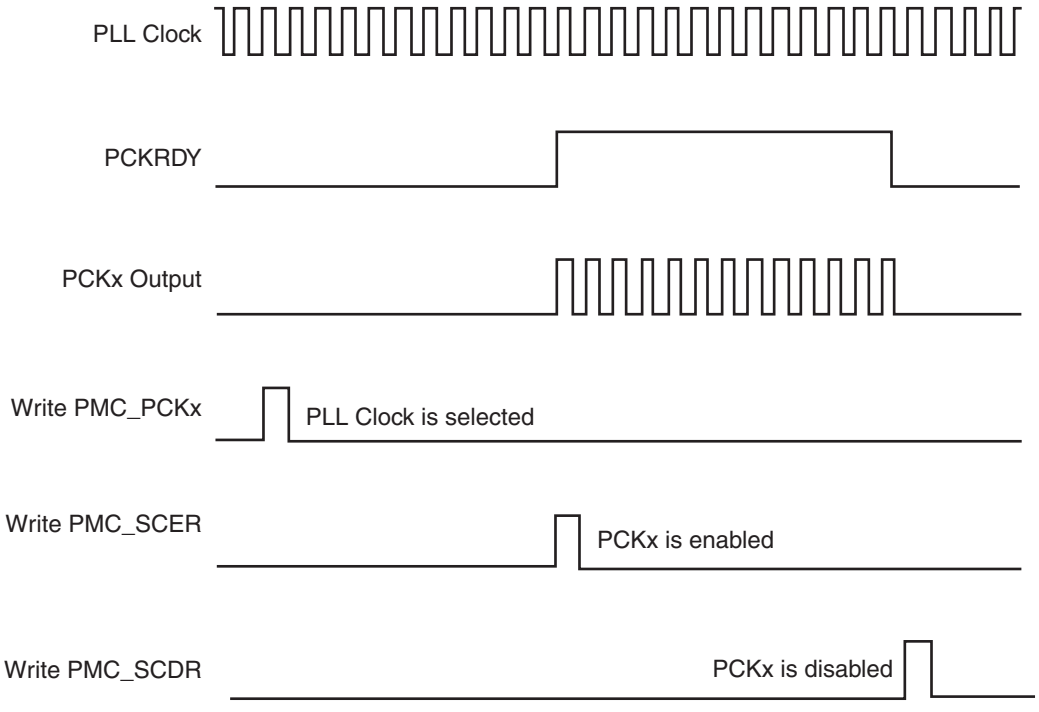


Figure 62. 可编程时钟输出编程



电源管理控制器 (PMC) 用户接口

Table 48. 电源管理控制器 (PMC) 寄存器映射

偏移	寄存器	名称	访问类型	复位值
0x0000	系统时钟使能寄存器	PMC_SCER	只写	–
0x0004	系统时钟禁用寄存器	PMC_SCDR	只写	–
0x0008	系统时钟状态寄存器	PMC _SCSR	只读	0x01
0x000C	保留	–	–	–
0x0010	外设时钟使能寄存器	PMC _PCER	只写	–
0x0014	外设时钟禁用寄存器	PMC_PCDR	只写	–
0x0018	外设时钟状态寄存器	PMC_PCSR	只读	0x0
0x001C	保留	–	–	–
0x0020	主振荡器寄存器	CKGR_MOR	读 / 写	0x0
0x0024	主时钟频率寄存器	CKGR_MCFR	只读	-
0x0028	保留	–	–	–
0x002C	PLL 寄存器	CKGR_PLLR	读 / 写	0x3F00
0x0030	主机时钟寄存器	PMC_MCKR	读 / 写	0x0
0x0038	保留	–	–	–
0x003C	保留	–	–	–
0x0040	可编程时钟 0 寄存器	PMC_PCK0	读 / 写	0x0
0x0044	可编程时钟 1 寄存器	PMC_PCK1	读 / 写	0x0
...	...	...	...	...
0x0060	中断使能寄存器	PMC_IER	只写	--
0x0064	中断禁用寄存器	PMC_IDR	只写	--
0x0068	状态寄存器	PMC_SR	只读	0x18
0x006C	中断屏蔽寄存器	PMC_IMR	只读	0x0
0x0070 - 0x00FC	保留	–	–	–

PMC 系统时钟使能寄存器
寄存器名称： PMC_SCER
访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	PCK2	PCK1	PCK0
7	6	5	4	3	2	1	0
UDP	–	–	–	–	–	–	PCK

- **PCK: 处理器时钟使能**

0 = 无效。
1 = 使能处理器时钟。

- **UDP: USB 器件端口时钟使能**

0 = 无效。
1 = 使能 USB 器件端口 48 MHz 时钟。

- **PCKx: 可编程时钟 x 输出使能**

0 = 无效。
1 = 使能相应的可编程时钟输出。

PMC 系统时钟禁用寄存器
寄存器名称： PMC_SCDR
访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	PCK2	PCK1	PCK0
7	6	5	4	3	2	1	0
UDP	–	–	–	–	–	–	PCK

- **PCK: 处理器时钟禁用**

0 = 无效。

1 = 禁用处理器时钟，用来使处理器进入空闲模式。

- **UDP: USB 器件端口时钟禁用**

0 = 无效。

1 = 禁用 USB 器件端口 48 MHz 时钟。

- **PCKx: 可编程时钟 x 输出禁用**

0 = 无效。

1 = 禁用相应的可编程时钟输出。

PMC 系统时钟状态寄存器
寄存器名称： PMC_SCSR
访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	PCK2	PCK1	PCK0
7	6	5	4	3	2	1	0
UDP	–	–	–	–	–	–	PCK

- **PCK: 处理器时钟状态**

0 = 处理器时钟禁用。

1 = 处理器时钟使能。

- **UDP: USB 器件端口状态**

0 = USB 器件端口 48 MHz 时钟 (UDPCK) 禁用。

1 = USB 器件端口 48 MHz 时钟 (UDPCK) 使能。

- **PCKx: 可编程时钟 x 输出状态**

0 = 相应的可编程时钟输出禁用。

1 = 相应的可编程时钟输出使能。

PMC 外设时钟使能寄存器
寄存器名称： PMC_PCER
访问类型： 只写

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	–	–

- **PIDx: 外设时钟 x 使能**

0 = 无效。

1 = 使能相应外设时钟。

Note: 对外设 ID 编程不执行对 PMC 工作无影响。

PMC 外设时钟禁用寄存器
寄存器名称： PMC_PCDR
访问类型： 只写

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	–	–

- **PIDx: 外设时钟 x 禁用**

0 = 无效。

1 = 禁用相应外设时钟。

PMC 外设时钟状态寄存器
寄存器名称： PMC_PCSR
访问类型： 只读

31	30	29	28	27	26	25	24
PID31	PID30	PID29	PID28	PID27	PID26	PID25	PID24
23	22	21	20	19	18	17	16
PID23	PID22	PID21	PID20	PID19	PID18	PID17	PID16
15	14	13	12	11	10	9	8
PID15	PID14	PID13	PID12	PID11	PID10	PID9	PID8
7	6	5	4	3	2	1	0
PID7	PID6	PID5	PID4	PID3	PID2	–	–

- **PIDx: 外设时钟 x 状态**

0 = 相应外设时钟禁用。
1 = 相应外设时钟使能。

PMC 时钟发生器主振荡器寄存器
寄存器名称： CKGR_MOR
访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
OSCOUNT							
7	6	5	4	3	2	1	0
–	–	–	–	–	–	OSCBYPASS	MOSCEN

- **MOSCEN: 主振荡器使能**

晶体连接在 XIN 与 XOUT 间。
0 = 主振荡器禁用。
1 = 主振荡器使能。 OSCBYPASS 必须置为 0。

- **OSCBYPASS: 振荡器旁路**

0 = 无效。
1 = 主振荡器旁路。 MOSCEN 必须置为 0。外部时钟必须连接在 XIN 上。

- **OSCOUNT: 主振荡器启动时间**

指定慢时钟周期乘以 8 作为主振荡器启动时间。

PMC 时钟发生器主时钟频率寄存器
寄存器名称： CKGR_MCFR
访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	MAINRDY

15	14	13	12	11	10	9	8
MAINF							

7	6	5	4	3	2	1	0
MAINF							

- **MAINF: 主时钟频率**

给出 16 个慢时钟周期中主时钟周期数。

- **MAINRDY: 主时钟就绪**

0 = MAINF 值无效或主振荡器禁用。

1 = 主振荡器提前使能且 MAINF 值有效。

PMC 时钟发生器 PLL 寄存器
寄存器名称： CKGR_PLLR
访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	USBDIV		–	MUL		

23	22	21	20	19	18	17	16
MUL							

15	14	13	12	11	10	9	8
OUT		PLLCOUNT					

7	6	5	4	3	2	1	0
DIV							

在使用 PMC 前需检查 PLL 输入频率可能的限制及乘数。

- **DIV: 分频器**

DIV	分频器选择
0	分频器输出为 0
1	绕过分频器
2 - 255	分频器输出是所选时钟由 DIV 分频后的值

- **PLLCOUNT: PLL 计数器**

给定 CKGR_PLLR 写入后而 PMC_SR 中 LOCK 位置位前的慢时钟周期数。

- **OUT: PLL 时钟频率范围**

输出		PLL 时钟频率范围
0	0	参见产品手册的 DC 特性部分
0	1	保留
1	0	参见产品手册的 DC 特性部分
1	1	保留

- **MUL: PLL 乘法器**

0 = PLL 无效。

1 到 2047 = PLL 时钟频率为 PLL 输入频率与 MUL+ 1 的乘积。

- **USBDIV: USB 时钟分频**

USBDIV		USB 时钟分频
0	0	分频器输出为 PLL 时钟输出
0	1	分频器输出为 PLL 时钟输出除以 2
1	0	分频器输出为 PLL 时钟输出除以 4
1	1	保留

PMC 主机时钟寄存器
寄存器名称： PMC_MCKR
访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	PRES			CSS	

- **CSS: 主机时钟选择**

CSS		时钟源选择
0	0	慢时钟
0	1	主时钟
1	0	保留
1	1	PLL 时钟

- **PRES: 主机时钟预分频**

PRES			主机时钟
0	0	0	选定时钟
0	0	1	选定时钟除以 2
0	1	0	选定时钟除以 4
0	1	1	选定时钟除以 8
1	0	0	选定时钟除以 16
1	0	1	选定时钟除以 32
1	1	0	选定时钟除以 64
1	1	1	保留

PMC 可编程时钟寄存器
寄存器名称： PMC_PCKx
访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	PRES			CSS	

- **CSS: 主机时钟选择**

CSS		时钟源选择
0	0	慢时钟
0	1	主时钟
1	0	保留
1	1	PLL 时钟

- **PRES: 可编程时钟预分频**

PRES			主机时钟
0	0	0	选定时钟
0	0	1	选定时钟除以 2
0	1	0	选定时钟除以 4
0	1	1	选定时钟除以 8
1	0	0	选定时钟除以 16
1	0	1	选定时钟除以 32
1	1	0	选定时钟除以 64
1	1	1	保留

PMC 中断使能寄存器
寄存器名称： PMC_IER
访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	PCKRDY2	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	LOCK	–	MOSCS

- **MOSCS: 主振荡器状态中断使能**
- **LOCK: PLL 锁定中断使能**
- **MCKRDY: 主机时钟就绪中断使能**
- **PCKRDYx: 可编程时钟就绪 x 中断使能**

0 = 无效。
1 = 使能相应中断。

PMC 中断禁用寄存器
寄存器名称： PMC_IDR
访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	PCKRDY2	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	LOCK	–	MOSCS

- **MOSCS: 主振荡器状态中断禁用**
- **LOCK: PLL 锁定中断禁用**
- **MCKRDY: 主机时钟就绪中断禁用**
- **PCKRDYx: 可编程时钟就绪 x 中断禁用**

0 = 无效。
1 = 禁用相应中断。

PMC 状态寄存器
寄存器名称： PMC_SR
访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	PCKRDY2	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	LOCK	–	MOSCS

- **MOSCS: MOSCS 标志状态**

0 = 主振荡器不稳定。

1 = 主振荡器稳定。

- **LOCK: PLL 锁定状态**

0 = PLL 未锁定。

1 = PLL 锁定。

- **MCKRDY: 主机时钟状态**

0 = 主机时钟未就绪。

1 = 主机时钟就绪。

- **PCKRDYx: 可编程时钟就绪状态**

0 = 可编程时钟 x 未就绪。

1 = 可编程时钟 x 就绪。

PMC 中断屏蔽寄存器
寄存器名称： PMC_IMR
访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	PCKRDY2	PCKRDY1	PCKRDY0
7	6	5	4	3	2	1	0
–	–	–	–	MCKRDY	LOCK	–	MOSCS

- **MOSCS: 主振荡器状态中断屏蔽**
- **LOCK: PLL 锁定中断屏蔽**
- **MCKRDY: 主机时钟就绪中断屏蔽**
- **PCKRDYx: 可编程时钟就绪 x 中断屏蔽**

0 = 相应中断使能。
1 = 相应中断禁用。

调试单元 (DBGU)

概述

调试单元为处理器访问基于 Atmel 的 ARM 内核系统所有调试功能提供了一个单入口点。

调试单元的 2 引脚 UART 可用于多种调试与追踪目的，它为 in-situ 编程方案及调试监控通信提供了理想媒介。此外，与两个外设数据控制器连接的通道允许进行批处理以降低处理器时间。

调试单元也可使得由 ARM 处理器板上仿真器提供的调试通信通道 (DCC) 信号对软件可见。这些信号指示 DCC 读与写寄存器状态并产生一个 ARM 处理器中断，以使对 DCC 处理在中断控制下进行。

芯片标识符寄存器用来辨别器件及其版本。这些寄存器中有片上存储器大小与类型，及内置外设。

最后，调试单元还有强制 NTRST 功能，使得软件可决定是否阻止通过板上仿真器访问系统。这允许将保护代码存于 ROM 中。

方框图

Figure 63. 调试单元功能框图

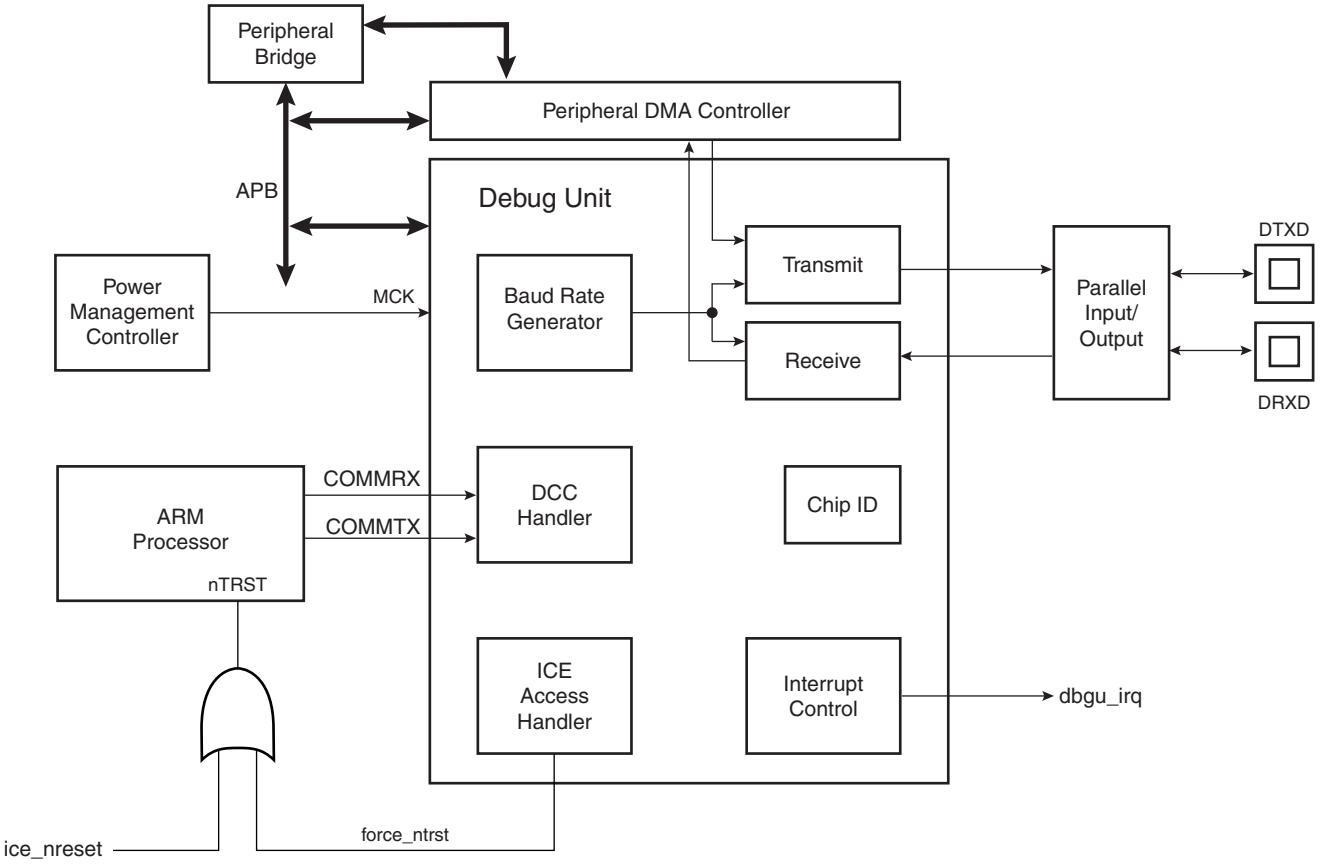
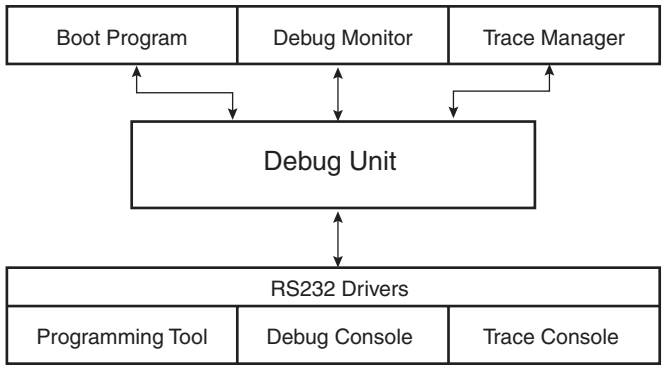


Table 49. 调试单元引脚说明

引脚名称	说明	类型
DRXD	调试接收数据	输入
DTXD	调试发送数据	输出

Figure 64. 调试单元应用示例



附属产品

I/O 线

根据产品集成，调试单元引脚可复用为 PIO 线。此时，程序员必须首先配置相应的 PIO 控制器以使能调试单元的 I/O 线操作。

电源管理

根据产品集成，调试单元时钟可由电源管理控制器来控制。此时，程序员必须首先配置 PMC 以使能调试单元时钟。通常此时的外设标识符为 1。

中断源

根据产品集成，调试单元中断线与高级中断控制器中的一个中断源连接。在配置调试单元前中断处理请求对 AIC 编程。通常，调试单元中断线与 AIC 中断源 1 连接，可共享实时时钟、系统定时器中断线及其它系统外设中断，见 Figure 63。为此程序员在触发源 1 时确定中断源。

UART 操作

调试单元作为 UART 运行时 (仅在异步模式下) 只支持 8 位字符处理 (有奇偶校验位)。没有时钟引脚。

调试单元的 UART 由独立工作的接收器与发送器及一个通用波特率发生器组成。计数器超时与发送器时间保护不执行。但是，所有执行特性与标准 USART 兼容。

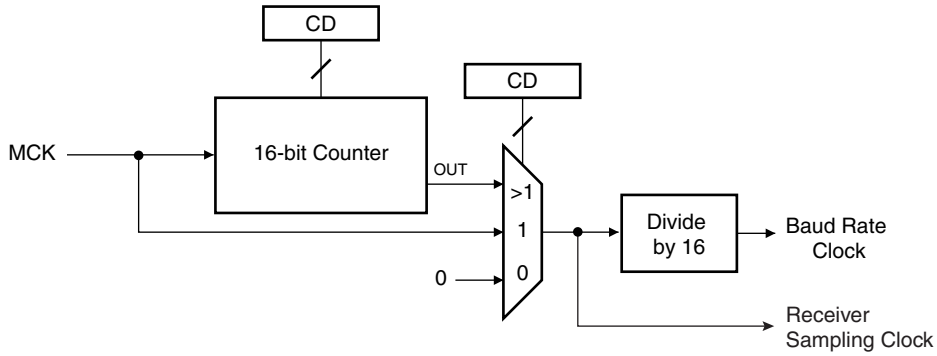
波特率发生器

波特率发生器向发送器与接收器提供名为波特率时钟的位周期时钟。

波特率时钟为主机时钟与 16 倍 DBGU_BRGR(波特率发生器寄存器)CD 值相除后的结果。若 DBGU_BRGR置为0 ,波特率时钟禁用且调试单元的UART无效。最大允许的波特率为主机时钟的 16 分频。最小允许的波特率为主机时钟的 16 x 65536 分频。

$$\text{Baud Rate} = \frac{\text{MCK}}{16 \times \text{CD}}$$

Figure 65. 波特率发生器



接收器

接收器复位，使能与禁用

器件复位后 ,调试单元接收器禁用 ,在使用前必须使能。接收器可通过在控制寄存器 DBGU_CR 的 RXEN 位写入 1 使能。该命令后，接收器开始寻找起始位。

程序员可在 DBGU_CR 寄存器的 RXDIS 写 1 来禁用接收器。若接收器正等待起始位，则立即停止。但若接收器已检测到起始位并正在接收数据，则将等待停止位出现后再停止。

程序员也可写 DBGU_CR 寄存器 RSTRX 位为 1 将接收器置于复位状态。此时，接收器立即停止当前操作并将其禁用。若当处理数据时设置 RSTRX，数据将丢失。

开始检测与数据采样

调试单元只支持异步操作，且仅影响接收器。调试单元接收器通过对 DRXD 信号采样来检测接收字符起始位。若采样时钟超过 7 个周期，采样时钟为 16 倍波特率， DRXD 上为低表示检测

到有效启动位。因此检测到一个比 7/16 长的位周期将视为有效起始位。等于或短于 7/16 将视为无效，接收器继续等待有效起始位。

检测到有效起始位后，接收器在各位理论中点处对 DRXD 采样。假设每位持续 16 个采样时钟周期 (1 位周期) 所以采样点是启动该位后第 8 个周期 (0.5 位周期)。第一个采样点是检测到启动位下降沿后的第 24 个周期 (1.5 位周期)。

每位在前一位 16 周期 (1 位周期) 后采样。

Figure 66. 起始位检测

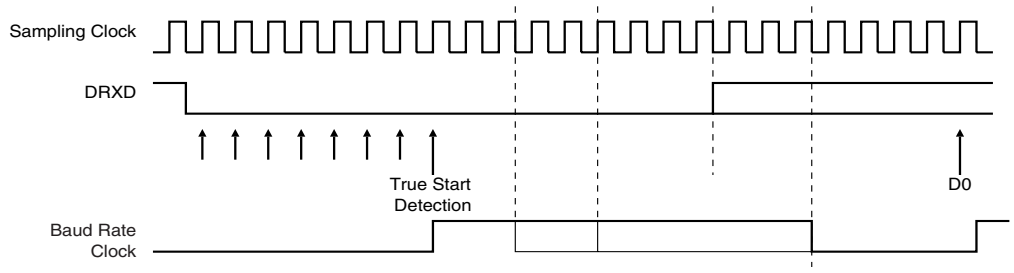
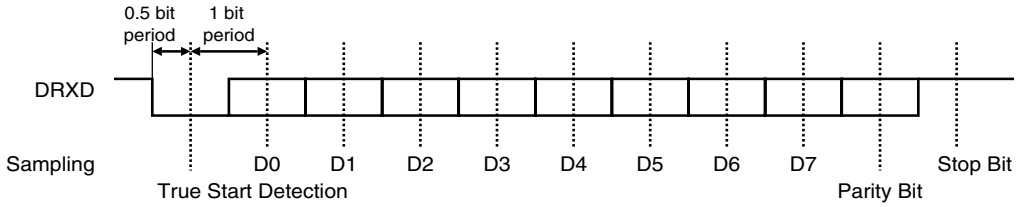


Figure 67. 字符接收

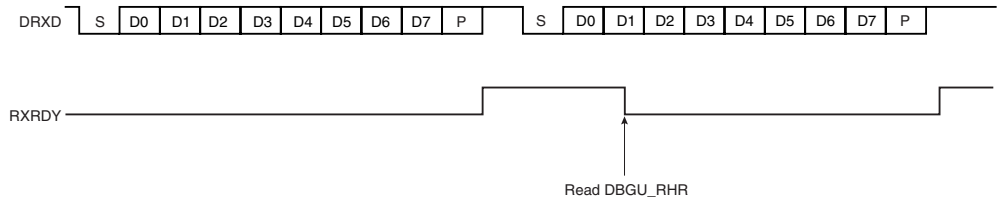
Example: 8-bit, parity enabled 1 stop



接收器就绪

接收一个完整字符后，传输到 DBGU_RHR 并将 DBGU_SR(状态寄存器) 中的 RXRDY 状态位置位。当读取接收保持寄存器 DBGU_RHR 时， RXRDY 位自动清除。

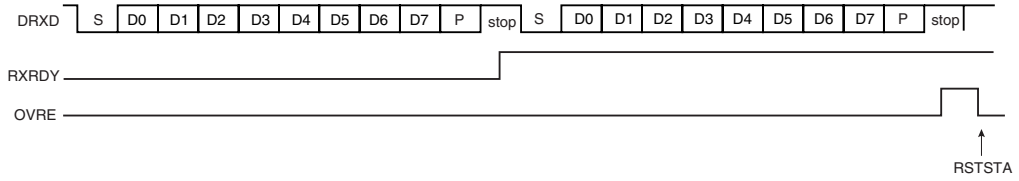
Figure 68. 接收器就绪



接收器溢出

上次传输后，若 DBGU_RHR 仍未被软件或外设数据控制器读取， RXRDY 位仍将置位并开始接收新字符，DBGU_SR 中的 OVRE 状态位置位。当软件将 DBGU_CR 的位 RSTSTA(复位状态) 写为 1 时， OVRE 位清除。

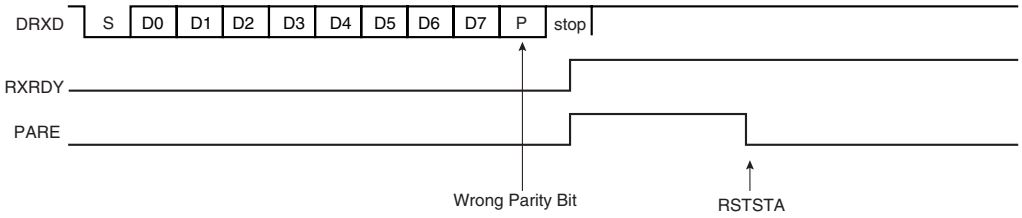
Figure 69. 接收器溢出



奇偶校验位错误

每次接收到字符后，接收器在 DBGU_MR 中的 PAR 域计算接收数据比特的奇偶校验。然后将计算结果与接收到的奇偶校验位比。若不同，在设置 RXRDY 的同时设置 DBGU_SR 中的奇偶错误校验位 PARE。当控制寄存器 DBGU_CR 的 RSTSTA 位写入 1，奇偶校验位清除。若在复位状态命令写入前收到新的字符， PARE 位仍为 1。

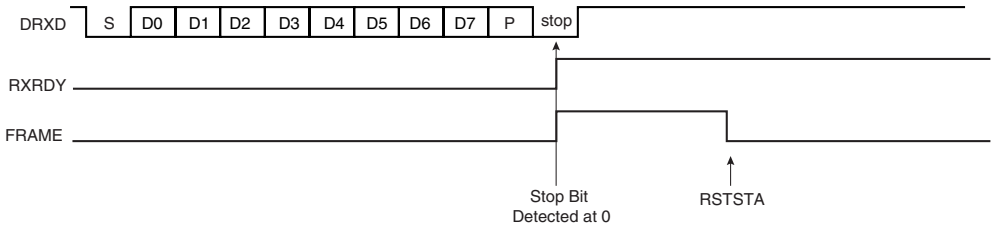
Figure 70. 奇偶校验错误



接收器帧错误

当检测到启动位，所有数据采样后产生一个字符接收信号。停止位也将采样，当检测到它为 0 时，DBGU_SR 中的 FRAME (帧错误) 位置位，同时将 RXRDY 位置位。FRAME 将保持为高直到寄存器 DBGU_CR 的 RSTSTA 位写入 1。

Figure 71. 接收器帧错误



发送器

发送器复位，使能与禁用

器件复位后，调试单元发送器禁用，在使用前必须使能。发送器可通过在控制寄存器 DBGU_CR 的 TXEN 位写入 1 使能。该命令后，发送器在开始工作前等待在发送保持寄存器 DBGU_THR 中写入符号。

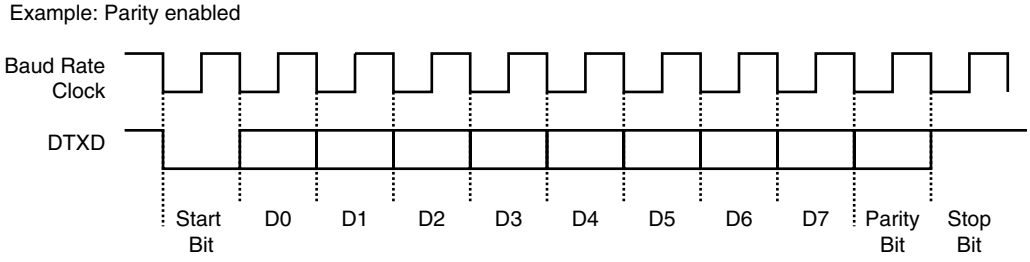
程序员可在 DBGU_CR 寄存器的 TXDIS 写 1 来禁用发送器。若发送器未工作，则立即停止。但若已有字符写入移位寄存器或已写入发送保持寄存器，则在字符发送完成后再停止。

程序员也可写DBGU_CR寄存器RSTRX位为1将发送器置于复位状态。此时，发送器立即停止。

发送格式

调试单元以波特率时钟速度驱动 DTXD 引脚。驱动总线由模式寄存器定义的格式决定，而数据保存在移位寄存器中。一个电平为 0 的起始位，然后是 8 个低位在先的数据位，一个可选的奇偶校验位及一个电平为 1 的停止位连续移出。模式寄存器 DBGU_MR 的 PARE 域定义是否将奇偶校验位移出。当奇偶校验位使能，可选用奇校验、偶校验或固定空或标记位。

Figure 72. 字符传输

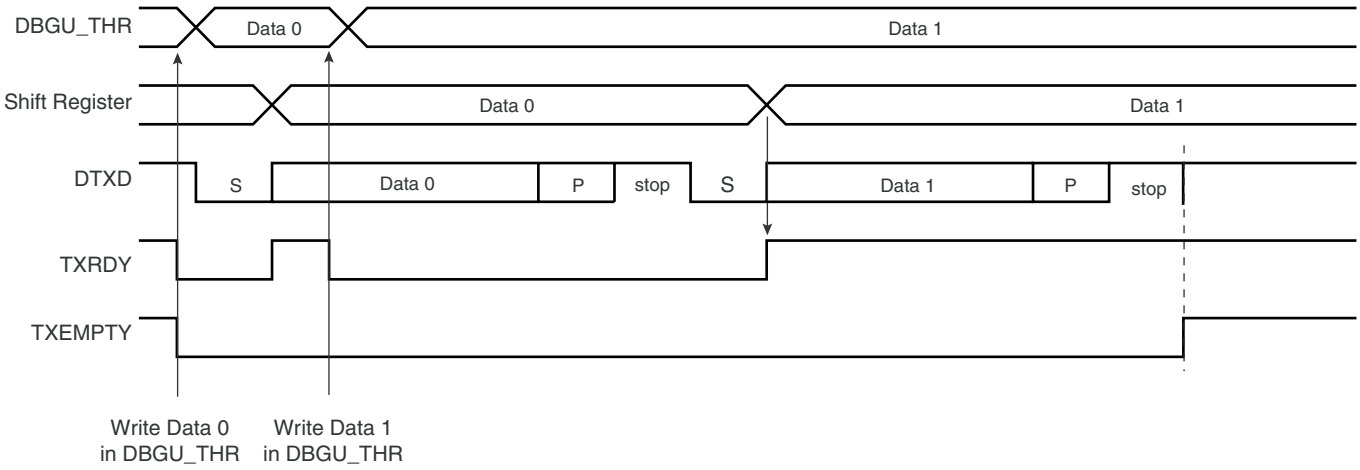


发送器控制

当发送器使能，DBGU_SR 中的 TXRDY(发送器就绪) 位置位。当程序员写发送保持寄存器 DBGU_THR 时，发送启动。在写字符发送完毕后数据由 DBGU_THR 发往移位寄存器。TXRDY 位保持为高直到第二个字符写入 DBGU_THR。首字符完成后，写入 DBGU_THR 的尾字符送入移位寄存器，且 TXRDY 重新拉高，此时保持寄存器为空。

当移位寄存器与DBGU_THR均为空时，即写入DBGU_THR中的所有字符均以处理，TXEMPTY 位在最后一个停止位结束后拉高。

Figure 73. 发送器控制



外设数据控制器

调试单元 UART 的发送器和接收器通常与外设数据控制器 (PDC) 通道连接。

外设数据控制器通道通过映射到调试单元接口偏移地址为 0x100 的寄存器编程。状态变化显示在调试状态寄存器 DBGU_SR 中并能产生一个中断。

RXRDY位触发PDC通道接收器数据传输。因此在DBGU_RHR中读取数据。TXRDY位触发PDC通道发送器数据传输。因此在 DBGU_THR 中写入数据。

测试模式

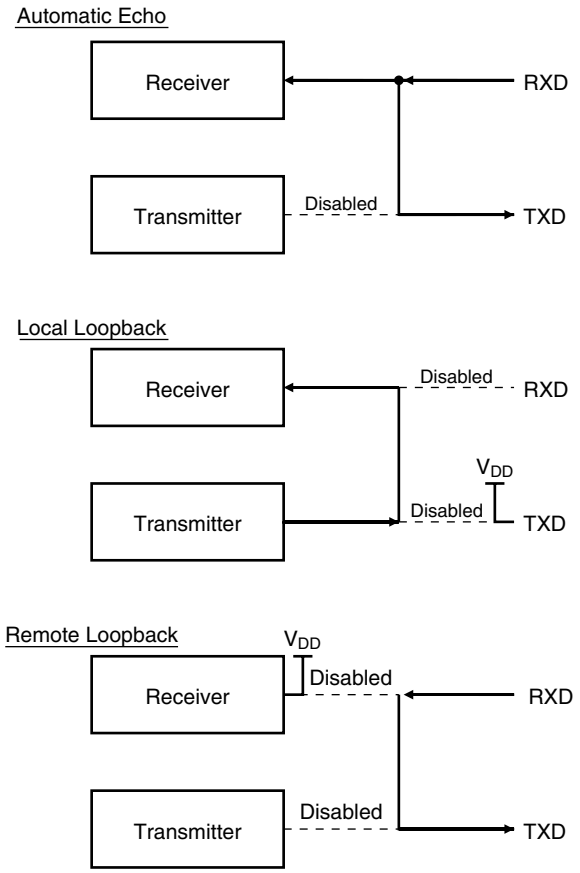
调试单元支持三种测试模式。由模式寄存器 DBGU_MR 的 CHMODE (通道模式) 域对使用哪种模式进行编程。

自动回应模式允许位转发。当 DRXD 线收到一个比特后，发送到 DTXD 线。发送器工作正常，但对 DTXD 线没有影响。

本地回环模式允许接收发送的字符。 DTXD 与 DRXD 引脚未用，发送器输出与接收器输入连接。与空闲状态相同， DRXD 引脚电平无影响， DTXD 线保持高电平。

远程回环模式直接将 DRXD 引脚与 DTXD 线连接。发送器与接收器禁用。该模式允许位重发。

Figure 74. 测试模式



调试通信通道支持

调试单元处理来自 ARM 处理器调试通信通道的 COMMRX 与 COMMTX 信号，用板上仿真器驱动。

调试通信通道包含两个由 JTAG 边 ICE 断路器及 ARM 处理器中协处理器 0 访问的寄存器。

下面的指令用来对调试通信通道进行读写：

```
MRC     p14, 0, Rd, c1, c0, 0
```

将调试通信数据寄存器中的内容返回 Rd

```
MCR     p14, 0, Rd, c1, c0, 0
```

将 Rd 中值写入调试通信数据写寄存器。

COMMRX 与 COMMTX 位，分别表示读寄存器已由调试器写入但仍未被处理器读出，及写寄存器已由处理器写入但仍未被调试器读出，位于状态寄存器 DBGU_SR 的最高两位。这两位可产生一个中断。该特性允许中断下处理运行于目标系统中的调试监控器和调试器间的调试链接。

芯片标识符

调试单元有两个芯片标识寄存器：DBGU_CIDR (芯片 ID 寄存器) 与 DBGU_EXID (扩展 ID 寄存器)。两个寄存器中均为只读的硬件固化值。首个寄存器包括下列域：

- EXT - 表示用到了扩展标识寄存器
- NVPTYP 与 NVPSIZ - 标识内置非易失性存储器类型及其大小
- ARCH - 标识内置外设
- SRAMSIZ - 标识内置 SRAM 大小

- EPROC - 标识内置 ARM 处理器
- VERSION - 给出芯片型号

第二个寄存器是由器件决定的，若 EXT 为 0，则其值为 0。

ICE 访问限制

调试单元可阻止通过 ARM 处理器的 ICE 接口对系统进行访问。通过 NTRST(DBGU_FNR) 寄存器中的 ICE 接口插入信号 NTRST 实现该特性。对 FNTRST (强制 NTRST) 位写 1 将阻止 TAP 控制器的操作。

标准器件中，FNTRST 位复位为 0，因此不会阻止 ICE 访问。

该特性主要用于不愿其芯片代码可见的 ROM 中。

调试单元用户接口

Table 50. 调试单元存储器映射

偏移	寄存器	寄存器名称	访问类型	复位值
0x0000	控制寄存器	DBGU_CR	只写	–
0x0004	模式寄存器	DBGU_MR	读 / 写	0x0
0x0008	中断使能寄存器	DBGU_IER	只写	–
0x000C	中断禁用寄存器	DBGU_IDR	只写	–
0x0010	中断屏蔽寄存器	DBGU_IMR	只读	0x0
0x0014	状态寄存器	DBGU_SR	只读	–
0x0018	接收保持寄存器	DBGU_RHR	只读	0x0
0x001C	发送保持寄存器	DBGU_THR	只写	–
0x0020	波特率发生器寄存器	DBGU_BRGR	读 / 写	0x0
0x0024 - 0x003C	保留	–	–	–
0X0040	芯片 ID 寄存器	DBGU_CIDR	只读	–
0X0044	芯片 ID 扩展寄存器	DBGU_EXID	只读	–
0X0048	强制 NTRST 寄存器	DBGU_FNR	读 / 写	0x0
0x004C - 0x00FC	保留	–	–	–
0x0100 - 0x0124	PDC 域	–	–	–

调试单元控制寄存器

寄存器名称 : DBGU_CR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	RSTSTA
7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	–

- **RSTRX: 接收器复位**

0 = 无效。

1 = 接收器逻辑复位并禁用。若此时正接收字符，则停止接收。

- **RSTTX: 发送器复位**

0 = 无效。

1 = 发送器逻辑复位并禁用。若此时正发送字符，则停止发送。

- **RXEN: 接收器使能**

0 = 无效。

1 = 若 RXDIS 为 0，接收器使能。

- **RXDIS: 接收器禁用**

0 = 无效。

1 = 接收器禁用。若正在处理字符且 RSTRX 未置位，则在停止接收器前将字符处理完成。

- **TXEN: 发送器使能**

0 = 无效。

1 = 若 TXDIS 为 0，发送器使能。

- **TXDIS: 发送器禁用**

0 = 无效。

1 = 发送器禁用。若正在处理字符且已有字符写入 DBGU_THR 而 RSTTX 未置位，则在停止发送器前将两字符处理完成。

- **RSTSTA: 状态位复位**

0 = 无效。

1 = 将 DBGU_SR 寄存器中的状态位 PARE、 FRAME 及 OVRE 复位。

调试单元模式寄存器

寄存器名称 : DBGU_MR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CHMODE		–	–	PAR			–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **PAR: 校验类型**

PAR			校验类型
0	0	0	偶校验
0	0	1	奇校验
0	1	0	空号：校验强制为 0
0	1	1	标记：校验强制为 1
1	x	x	无奇偶校验

- **CHMODE: 通道模式**

CHMODE		模式说明
0	0	正常模式
0	1	自动回应
1	0	本地回环
1	1	远程回环

调试单元中断使能寄存器

寄存器名称： DBGU_IER

访问类型： 只写

31	30	29	28	27	26	25	24
COMMRX	COMMTX	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

- **RXRDY: 使能 RXRDY 中断**
- **TXRDY: 使能 TXRDY 中断**
- **ENDRX: 使能接收传输结束中断**
- **ENDTX: 使能发送结束中断**
- **OVRE: 使能溢出错误中断**
- **FRAME: 使能帧错误中断**
- **PARE: 使能奇偶校验错误中断**
- **TXEMPTY: 使能 TXEMPTY 中断**
- **TXBUFE: 使能缓冲器空中断**
- **RXBUFF: 使能缓冲器满中断**
- **COMMTX: 使能 COMMTX (来自 ARM) 中断**
- **COMMRX: 使能 COMMRX (来自 ARM) 中断**

0 = 无效。

1 = 使能相应中断。

调试单元中断禁用寄存器

寄存器名称 : DBGU_IDR

访问类型 : 只写

31	30	29	28	27	26	25	24
COMMRX	COMMTX	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

- **RXRDY: 禁用 RXRDY 中断**
- **TXRDY: 禁用 TXRDY 中断**
- **ENDRX: 禁用接收传输结束中断**
- **ENDTX: 禁用发送结束中断**
- **OVRE: 禁用溢出错误中断**
- **FRAME: 禁用帧错误中断**
- **PARE: 禁用奇偶校验错误中断**
- **TXEMPTY: 禁用 TXEMPTY 中断**
- **TXBUFE: 禁用缓冲器空中断**
- **RXBUFF: 禁用缓冲器满中断**
- **COMMTX: 禁用 COMMTX (来自 ARM) 中断**
- **COMMRX: 禁用 COMMRX (来自 ARM) 中断**

0 = 无效。

1 = 禁用相应中断。

调试单元中断屏蔽寄存器

寄存器名称： DBGU_IMR

访问类型： 只读

31	30	29	28	27	26	25	24
COMMRX	COMMTX	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

- **RXRDY: 屏蔽 RXRDY 中断**
- **TXRDY: 屏蔽 TXRDY 中断**
- **ENDRX: 屏蔽接收传输结束中断**
- **ENDTX: 屏蔽发送结束中断**
- **OVRE: 屏蔽溢出错误中断**
- **FRAME: 屏蔽帧错误中断**
- **PARE: 屏蔽奇偶校验错误中断**
- **TXEMPTY: 屏蔽 TXEMPTY 中断**
- **TXBUFE: 屏蔽缓冲器空中断**
- **RXBUFF: 屏蔽缓冲器满中断**
- **COMMTX: 屏蔽 COMMTX 中断**
- **COMMRX: 屏蔽 COMMRX 中断**

0 = 相应中断禁用。

1 = 相应中断使能。

调试单元状态寄存器

寄存器名称： DBGU_SR

访问类型： 只读

31	30	29	28	27	26	25	24
COMMRX	COMMTX	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	RXBUFF	TXBUFE	–	TXEMPTY	–
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	–	TXRDY	RXRDY

- **RXRDY: 接收器就绪**

0 = 上次读 DBGU_RHR 后未收到字符或接收器禁用。

1 = 至少收到一个完整的字符传输到 DBGU_RHR 并还未读。

- **TXRDY: 发送器就绪**

0 = 已将字符写入 DBGU_THR 但仍未发送到移位寄存器或发送器禁用。

1 = DBGU_THR 中没有未发送到移位寄存器中的字符。

- **ENDRX: 接收器传输结束**

0 = 接收器外设数据控制器通道中的传输结束信号无效。

1 = 接收器外设数据控制器通道中的传输结束信号有效。

- **ENDTX: 发送器传输结束**

0 = 发送器外设数据控制器通道中的传输结束信号无效。

1 = 发送器外设数据控制器通道中的传输结束信号有效。

- **OVRE: 溢出错误**

0 = 上次 RSTSTA 后无溢出错误。

1 = 上次 RSTSTA 后至少发生一次溢出错误。

- **FRAME: 帧错误**

0 = 上次 RSTSTA 后无帧错误。

1 = 上次 RSTSTA 后至少发生一次帧错误。

- **PARE: 奇偶校验错误**

0 = 上次 RSTSTA 后无奇偶校验错误。

1 = 上次 RSTSTA 后至少发生一次奇偶校验错误。

- **TXEMPTY: 发送器空**

0 = DBGU_THR 中有字符，或发送器正在处理字符或发送器禁用。

1 = DBGU_THR 中无字符且发送器没有处理字符。

- **TXBUFE: 发送缓冲器空**

0 = 来自发送器 PDC 通道的缓冲器空信号无效。

1 = 来自发送器 PDC 通道的缓冲器空信号有效。

- **RXBUFF: 接收缓冲器满**

0 = 来自接收器 PDC 通道的缓冲器满信号无效。

1 = 来自接收器 PDC 通道的缓冲器满信号有效。

- **COMMTX: 调试通信通道写状态**

0 = 来自 ARM 处理器的 COMMTX 无效。

1 = 来自 ARM 处理器的 COMMTX 有效。

- **COMMRX: 调试通信通道读状态**

0 = 来自 ARM 处理器的 COMMRX 无效。

1 = 来自 ARM 处理器的 COMMRX 有效。

调试单元接收器保持寄存器

寄存器名称： DBGU_RHR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
RXCHR							

- **RXCHR: 收到的字符**

若 RXRDY 置位，则为最后收到的字符。

调试单元发送保持寄存器

寄存器名称 : DBGU_THR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
TXCHR							

- **TXCHR: 要发送的字符**

若 TXRDY 未置位，为下一个要发送的字符。

调试单元波特率发生器寄存器

寄存器名称 : DBGU_BRGR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CD							
7	6	5	4	3	2	1	0
CD							

- **CD: 时钟分频器**

CD	波特率时钟
0	禁用
1	MCK
2 到 65535	MCK / (CD x 16)

调试单元芯片 ID 寄存器

寄存器名称： DBGU_CIDR

访问类型： 只读

31	30	29	28	27	26	25	24
EXT	NVPTYP			ARCH			
23	22	21	20	19	18	17	16
ARCH				SRAMSIZ			
15	14	13	12	11	10	9	8
NVPSIZ2				NVPSIZ			
7	6	5	4	3	2	1	0
EPROC			VERSION				

- **VERSION: 器件版本**
- **EPROC: 内置处理器**

EPROC			处理器
0	0	1	ARM946ES
0	1	0	ARM7TDMI
1	0	0	ARM920T
1	0	1	ARM926EJS

- **NVPSIZ: 非易失性程序存储器大小**

NVPSIZ				大小
0	0	0	0	无
0	0	0	1	8K 字节
0	0	1	0	16K 字节
0	0	1	1	32K 字节
0	1	0	0	保留
0	1	0	1	64K 字节
0	1	1	0	保留
0	1	1	1	128K 字节
1	0	0	0	保留
1	0	0	1	256K 字节
1	0	1	0	512K 字节
1	0	1	1	保留
1	1	0	0	1024K 字节
1	1	0	1	保留
1	1	1	0	2048K 字节
1	1	1	1	保留

- **NVPSIZ2: 第二个非易失性程序存储器大小**

NVPSIZ2				大小
0	0	0	0	无
0	0	0	1	8K 字节
0	0	1	0	16K 字节
0	0	1	1	32K 字节
0	1	0	0	保留
0	1	0	1	64K 字节
0	1	1	0	保留
0	1	1	1	128K 字节
1	0	0	0	保留
1	0	0	1	256K 字节
1	0	1	0	512K 字节
1	0	1	1	保留
1	1	0	0	1024K 字节
1	1	0	1	保留
1	1	1	0	2048K 字节
1	1	1	1	保留

- **SRAMSIZ: 内部 SRAM 大小**

SRAMSIZ				大小
0	0	0	0	保留
0	0	0	1	1K 字节
0	0	1	0	2K 字节
0	0	1	1	保留
0	1	0	0	保留
0	1	0	1	4K 字节
0	1	1	0	保留
0	1	1	1	160K 字节
1	0	0	0	8K 字节
1	0	0	1	16K 字节
1	0	1	0	32K 字节
1	0	1	1	64K 字节
1	1	0	0	128K 字节
1	1	0	1	256K 字节
1	1	1	0	96K 字节
1	1	1	1	512K 字节

- **ARCH: 结构标识符**

ARCH		结构
Hex	Bin	
0x40	0100 0000	AT91x40 系列
0x63	0110 0011	AT91x63 系列
0x55	0101 0101	AT91x55 系列
0x42	0100 0010	AT91x42 系列
0x92	1001 0010	AT91x92 系列
0x34	0011 0100	AT91x34 系列
0x70	0111 0000	AT91SAM7Sxx 与 AT91SAM7Axx 系列
0x71	0111 0001	AT91SAM7Xxx 系列
0x72	0111 0010	AT91SAM7Exx 系列
0x73	0111 0011	AT91SAM7Lxx 系列
0x19	0001 1001	AT91SAM9xx 系列

- **NVPTYP: 非易失性程序存储器类型**

NVPTYP			存储器
0	0	0	ROM
0	0	1	ROMless 或片上 Flash
1	0	0	SRAM 仿真 ROM
0	1	0	内置 Flash 存储器
0	1	1	ROM 与内置 Flash 存储器 NVPSIZ 为 ROM 大小 NVPSIZ2 为 Flash 大小

- **EXT: 扩展标志**

0 = 芯片 ID 在无扩展的单寄存器中定义。

1 = 存在扩展芯片 ID。

调试单元芯片 ID 扩展寄存器

寄存器名称 : DBGU_EXID

访问类型 : 只读

31	30	29	28	27	26	25	24
EXID							
23	22	21	20	19	18	17	16
EXID							
15	14	13	12	11	10	9	8
EXID							
7	6	5	4	3	2	1	0
EXID							

- **EXID: 芯片 ID 扩展**

若 DBGU_CIDR 中 EXT 位为 0，则其值为 0。

调试单元强制 NTRST 寄存器

寄存器名称 : DBGU_FNR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	FNTRST

- **FNTRST: 强制 NTRST**

0 = ARM 处理器的 TAP 控制器 NTRST 由 ice_nreset 信号驱动。

1 = ARM 处理器的 TAP 控制器 NTRST 为低。

并行输入 / 输出控制器 (PIO)

概述

并行输入 / 输出控制器 (PIO) 管理高达 32 全可编程输入 / 输出线。每个 I/O 线可作为通用功能 I/O 或分配给一个内置外设。这将确保产品引脚的有效优化。

每个 I/O 线均有一个与 32 位宽的用户接口的 32 位寄存器相关的位序号。

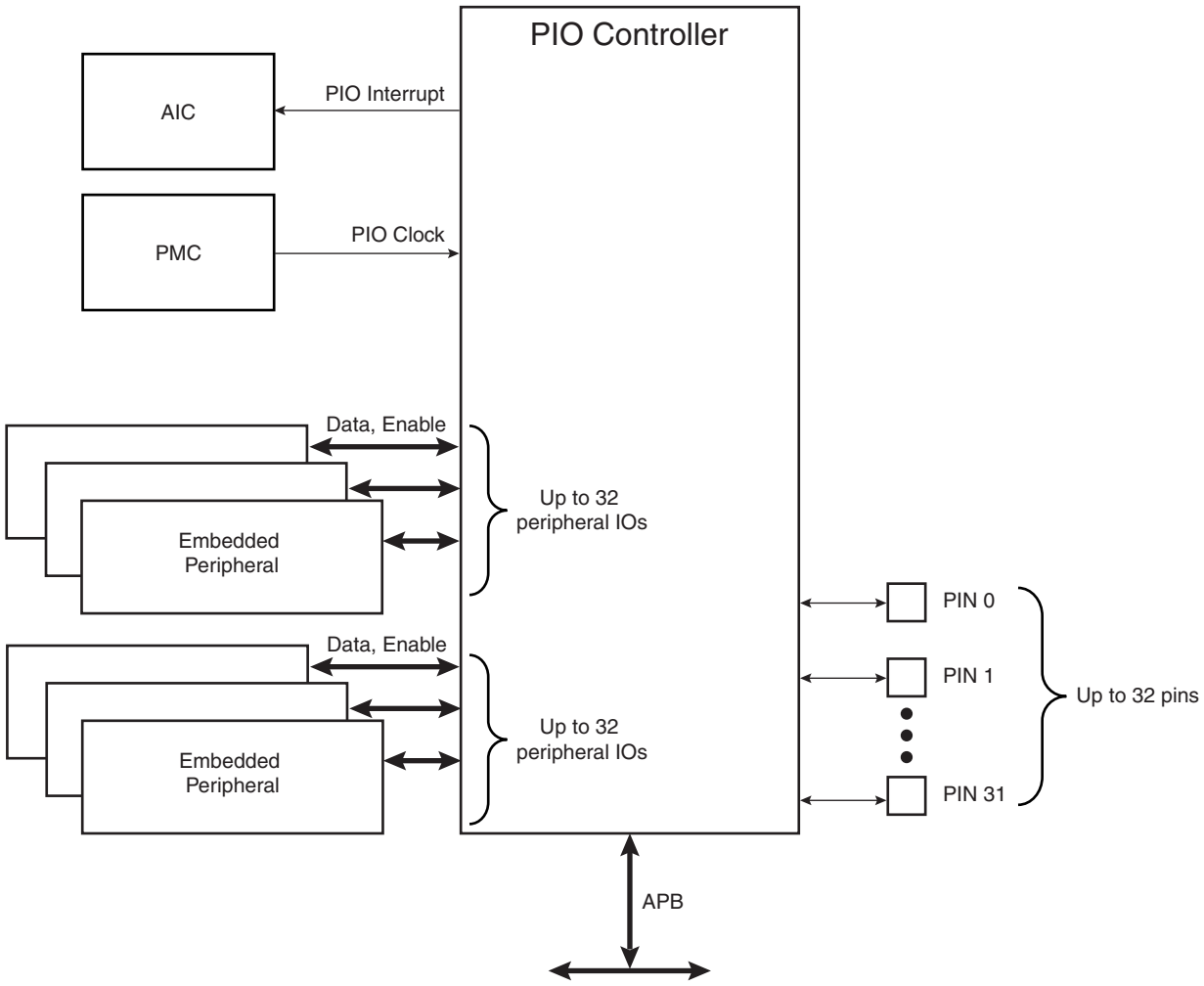
PIO 控制器的每个 I/O 线有如下特性：

- 任意 I/O 线上的输入改变中断将使能电平变化检测。
- 毛刺滤波器可拒绝低于 1.5 时钟周期的脉冲。
- 类似于开漏 I/O 线的多驱动能力。
- 控制 I/O 线上拉。
- 输入可见，输出控制。

PIO 控制器还有同步输出特性，可在单写操作中提供高达 32 位的数据输出。

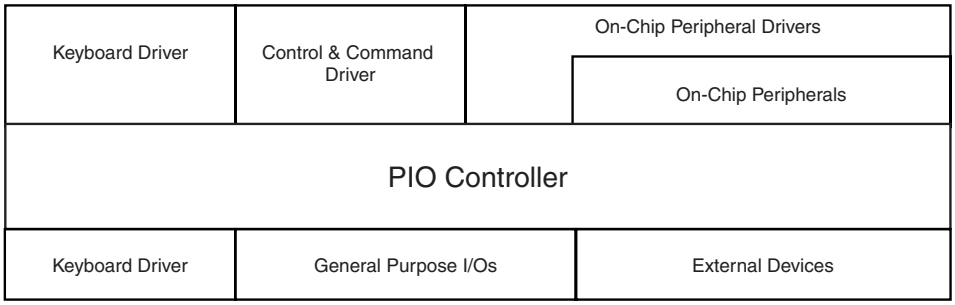
方框图

Figure 75. 方框图



应用框图

Figure 76. 应用框图



附属产品

引脚复用

每个引脚可配置为通用功能 I/O 线或与一个或两个外设 I/O 复用的 I/O 线。由于复用是硬件定义的，因此硬件设计师与程序员必须根据应用需要小心决定 PIO 控制器配置。若只作为通用功能 I/O 线使用，即没有与任何外设 I/O 复用，对分配给外设的 PIO 控制器编程无效，且仅 PIO 控制器可控制如何驱动引脚。

外部中断线

中断信号 FIQ 及 IRQ0 到 IRQn 一般通过 PIO 控制器复用。但是，由于 PIO 控制器对于输入无效且中断线 (FIQ 或 IRQ) 仅作为输入，因此不必为中断分配 I/O 线。

电源管理

电源管理控制器控制 PIO 控制器时钟以节省功耗。对用户接口寄存器写入时不需要将 PIO 控制器时钟使能。即配置 I/O 线 不需要将 PIO 控制器时钟使能。

但当时钟禁用时，PIO 控制器某些功能将不可用。输入变化中断与读引脚电平就需要时钟有效。

硬件复位后，默认将 PIO 时钟禁用。

在访问输入线信息前必须配置电源管理控制器。

中断产生

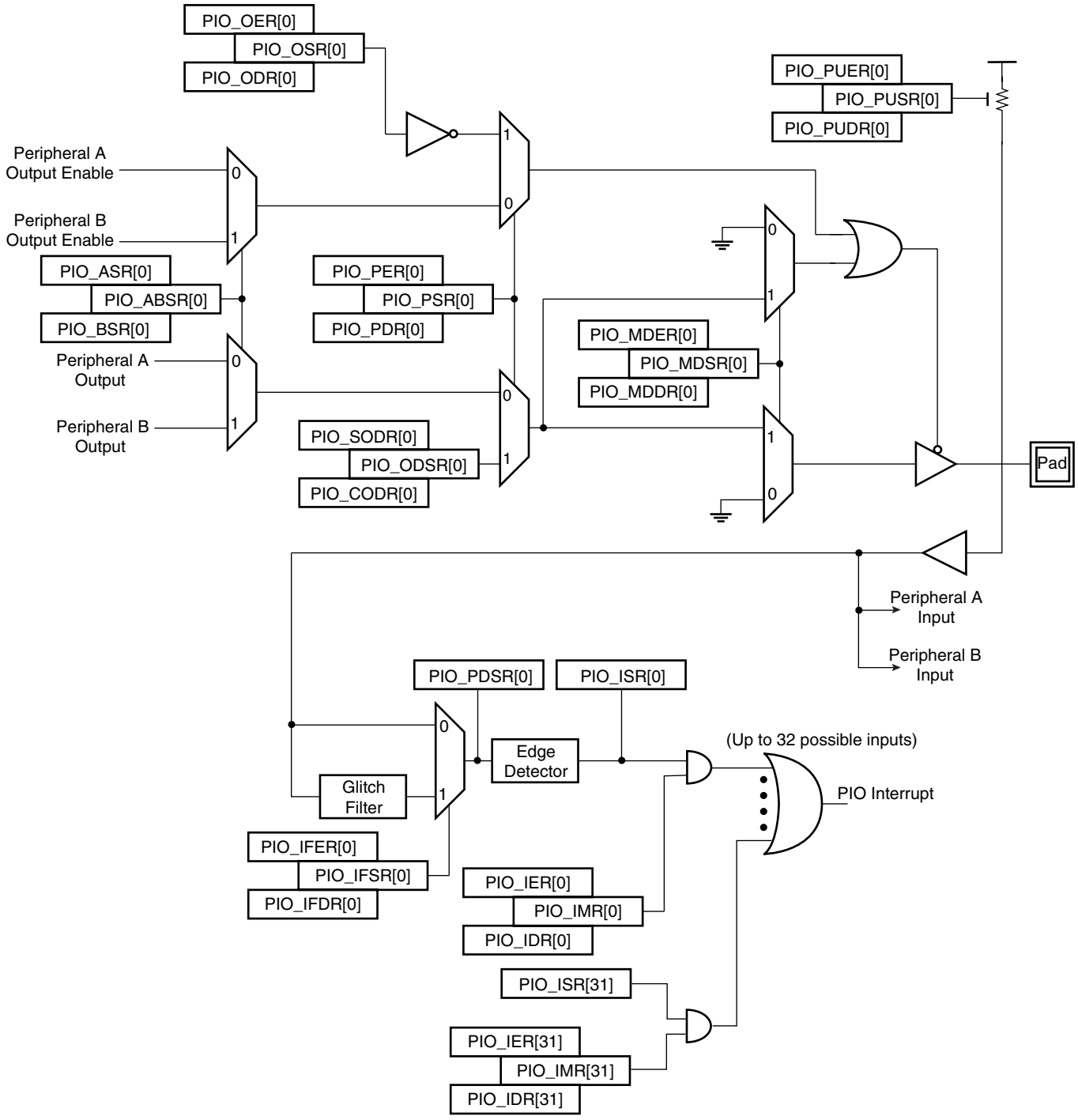
对于中断处理，认为 PIO 控制器为用户外设。即 PIO 控制器中断线连接在中断源 2 到 31 间。参见 PIO 控制器外设标识符以确定分配给 PIO 控制器的中断源。

只有当 PIO 控制器时钟使能时才能产生 PIO 控制器中断。

功能说明

PIO 控制器有多达 32 的全可编程 I/O 线。Figure 77 给出了大部分与每个 I/O 相关的控制逻辑。

Figure 77. I/O 线控制逻辑



上拉电阻控制

每个 I/O 线都有一个内置上拉电阻。阻值约为 100 kΩ (详见产品电气特性)。通过对 PIO_PUER (上拉使能电阻) 及 PIO_PUDR (上拉禁用电阻) 写入可使能或禁用上拉电阻。对这些寄存器写入的结果是置位或清零 PIO_PUSR(上拉状态寄存器) 中的相应位。 PIO_PUSR 读出值为 1 表示上拉禁用，读出值为 0 则表示上拉使能。

对上拉电阻的控制与 I/O 线配置无关。

复位后，所有上拉使能，即 PIO_PUSR 复位值为 0x0。

I/O 线或外设功能选择

当引脚复用一或两个外设功能，通过 PIO_PER (PIO 使能寄存器) 及 PIO_PDR (PIO 禁用寄存器) 控制功能选择。PIO_PSR (PIO 状态寄存器) 为寄存器置位或清零的结果，并用来指示引脚是由相应的外设还是由 PIO 控制器来控制。值为 0 时表示引脚由 PIO_ABSR (AB 选择状态寄存器) 相应的片上外设选择来控制；值为 1 表示由 PIO 控制器来控制。

若引脚作为通用功能 I/O 线使用 (未与片上外设复用) ,PIO_PER 与 PIO_PDR 无效且 PIO_PSR 相应位返回 1。

通常情况下，复位后， I/O 线由 PIO 控制器来控制，即 PIO_PSR 复位值为 1。但某些情况下，PIO要由外设控制(如复位后存储器片选必须置为无效或外部存储器导出时地址线必须为低)。因此， PIO_PSR 复位值由产品定义，取决于器件复用。

外设 A 或 B 选择

PIO控制器可在单引脚上提供两外设功能复用。通过对PIO_ASR (A 选择寄存器)与PIO_BSR (B 选择寄存器) 写入来选择。 PIO_ABSR 指出当前选择的外设线。对于每个引脚，相应位为 0 表示选择外设 A ；相应位为 1 表示选择外设 B。

注意外设线 A 与 B 复用仅对输出线有影响。外设输入线只与引脚输入连接。

复位后 PIO_ABSR 为 0，即表示所有 PIO 线分配给外设 A。但是在 I/O 线模式下， PIO 控制器复位后，外设 A 通常不会驱动该引脚。

不管引脚配置，对 PIO_ASR 及 PIO_BSR 写入时将影响 PIO_ABSR 值。但是在给引脚分配外设功能时除需写 PIO_PDR 外还需写入相应外设选择寄存器 (PIO_ASR 或 PIO_BSR)。

输出控制

当 I/O 线分配为外设功能后，即 PIO_PSR 相应位为 0，由外设控制 I/O 线驱动。外设 A 或 B 由 PIO_ABSR 值确定是否驱动引脚。

当 I/O 线由 PIO 控制器控制，引脚可配置为驱动。通过写 PIO_OER (输出使能寄存器) 及 PIO_PDR (输出禁用寄存器)实现。写操作结果由PIO_OSR (输出状态寄存器)中得到。当该寄存器中位值为 0，相应 I/O 线只作为输入使用；当该位值为 1，相应 I/O 线由 PIO 控制器驱动。

通过写 PIO_SODR (置位输出数据寄存器) 与 PIO_CODR (清零输出数据寄存器) ,可将 I/O 改为电平驱动。写操作对 PIO_ODSR (输出数据状态寄存器) 分别置位与清零，表示在 I/O 线上的数据驱动。不管引脚配置为 PIO 控制器控制还是分配给外设功能，写 PIO_OER 与 PIO_ODR 将导致 PIO_OSR 变化。这使得可通过 PIO 控制器管理优先设置 I/O 线。

类似的，写 PIO_SODR 与 PIO_CODR 将影响 PIO_ODSR。这对于定义 I/O 线第一级驱动非常重要。

同步数据输出

使用几个 PIO 控制所有并行总线需要两个连续的对 PIO_SODR 与 PIO_CODR 寄存器的写操作。这可能会出现未知瞬间值。 PIO 控制器通过对 PIO_ODSR 单写访问，提供了对 PIO 输出的直接控制。只有未被 PIO_OSWSR (输出写状态寄存器) 屏蔽的位可写入。通过写 PIO_OWER(输出写使能寄存器)将PIO_OWSR中屏蔽位置位，而对PIO_OWDR (输出写禁用寄存器) 写将对屏蔽位清零。

复位后，由于 PIO_OWSR 复位值为 0x0，所有 I/O 线的同步数据输出禁用。

多驱动控制 (开漏)

通过使用多驱动控制，每个 I/O 可在开漏时独立编程。该特性允许多个驱动器连接到 I/O 线上，I/O线仅可由每个器件驱动为低。一般需要一个外部上拉电阻 (或使能内部上拉电阻) 以确保线上高电平。

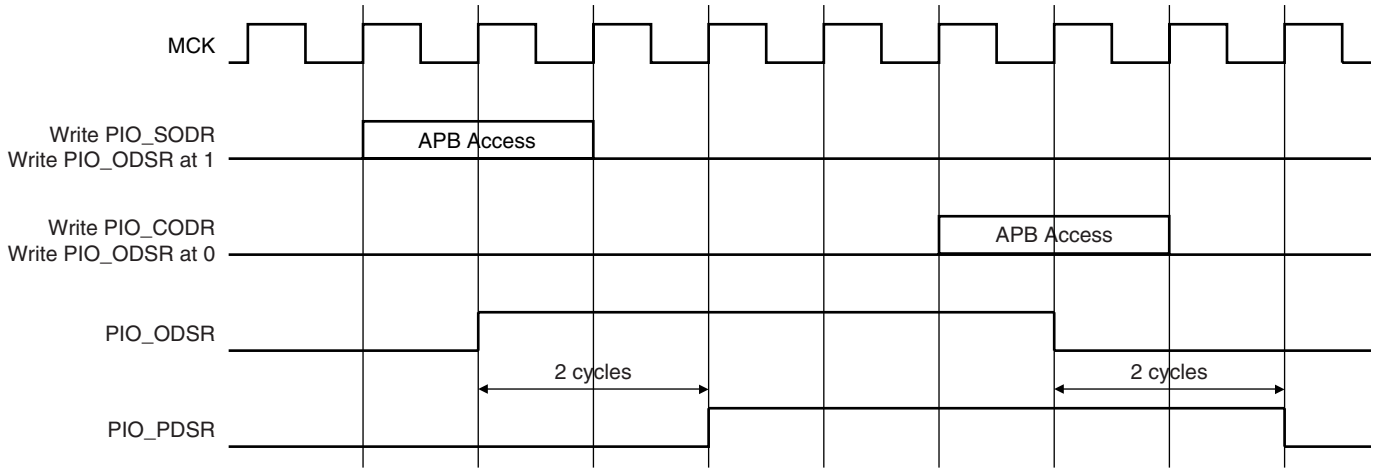
多驱动特性由 PIO_MDER (多驱动器使能寄存器) 及 PIO_MDDR (多驱动器禁用寄存器) 来控制。可选择多驱动的 I/O 线由 PIO 控制还是分配给外设功能。 PIO_MDSR (多驱动状态寄存器) 表示配置引脚支持外部驱动器。

复位后，由于 PIO_MDSR 复位值为 0x0，所有引脚多驱动特性禁用。

输出线时序

Figure 78 给出写 PIO_SODR 或 PIO_CODR 或直接写 PIO_ODSR 的驱动输出。最后一种情况只有在 PIO_OWSR 中相应位置位时才有效。 Figure 78 给出了 PIO_PDSR 反馈有效的时刻。

Figure 78. 输出线时序



输入

每个 I/O 线电平可通过 PIO_PDSR (外设数据状态寄存器) 读出。该寄存器指示 I/O 线电平，不管它是仅作为输入还是由 PIO 控制器或外设驱动。

对 I/O 线电平读取时需要将 PIO 控制器时钟使能，否则 PIO_PDSR 读到的是时钟禁用时的 I/O 线电平。

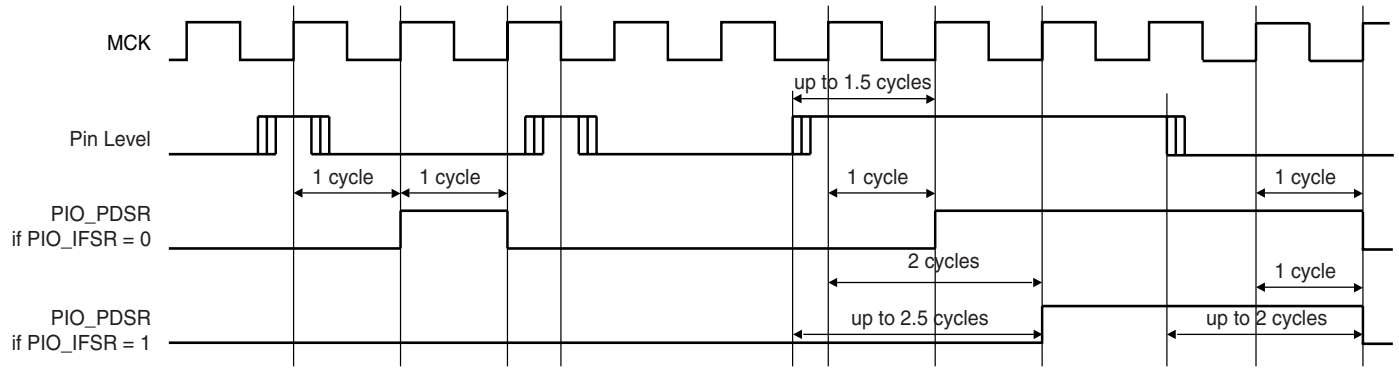
输入毛刺滤波

每个 I/O 线可选的输入毛刺滤波器可独立编程。当毛刺滤波器使能，自动滤除小于 1/2 主机时钟 (MCK) 周期的毛刺，而当大于 1 个主机时钟周期的信号将通过。对于在 1/2 到 1 个主机时钟周期间的信号则由其出现的精确时间决定是否能通过。因此对于信号必须超过 1 个主机时钟周期，而毛刺则要小于 1/2 主机时钟周期。若在上升沿前引脚电平变化，则滤波器引入 1 个主机时钟周期延迟。但若在下降沿前出现引脚电平变化，则不会有延迟，详见 Figure 79。

毛刺滤波器由 PIO_IFER (输入滤波器使能寄存器)、 PIO_IFDR (输入滤波器禁用寄存器) 及 PIO_IFSR (输入滤波器状态寄存器) 控制。分别对 PIO_IFER 及 PIO_IFDR 写入将置位或清零 PIO_IFSR 中位。最后一个寄存器使能 I/O 线上的毛刺滤波器。

当毛刺滤波器使能，不会修改外设输入。它仅在读 PIO_PDSR 值及输入改变中断检测中起作用。毛刺滤波器要求 PIO 控制器时钟使能。

Figure 79. 输入毛刺滤波器时序



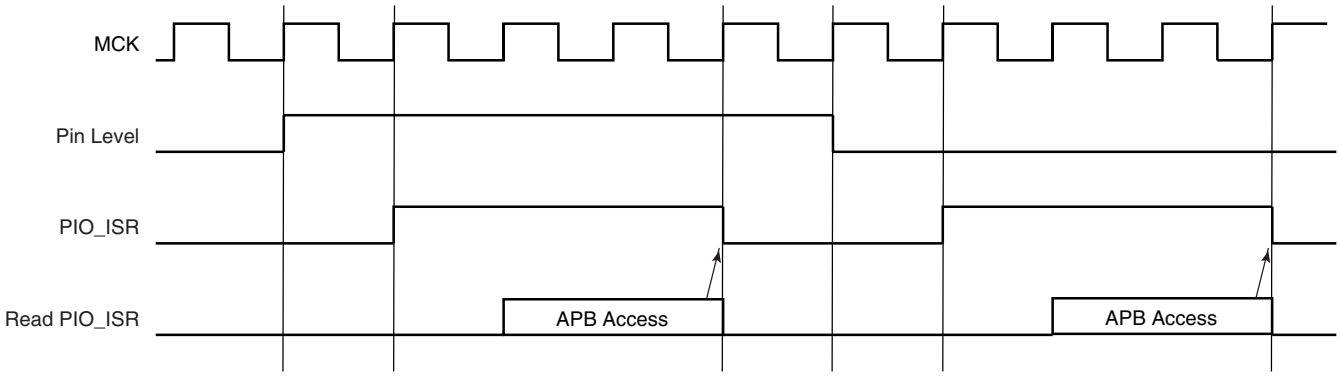
输入变化中断

当 PIO 控制器检测到 I/O 线上输入变化时，它可编程产生中断。输入变化中断由 PIO_IER（中断使能寄存器）与 PIO_IDR（中断禁用寄存器）的写入来控制，两寄存器分别通过置位与清零 PIO_IMR（中断屏蔽寄存器）中的相应位来使能或禁用输入变化中断。由于输入变化检测只能通过比较连续两个输入采样值得到，因此 PIO 控制器时钟必须使能。不管 I/O 线如何配置，输入变化中断有效 。

当检测到 I/O 线上的输入变化，PIO_ISR（中断状态寄存器）中的相应位置位。若 PIO_IMR 相应位置位，插入 PIO 控制器中断。32 通道中断信号线或后产生一个单中断信号，送往高级中断控制器。

当软件读取PIO_ISR时，所有中断自动清零。即当读取PIO_ISR时，所有挂起的中断必须处理。

Figure 80. 输入变化中断时序



I/O 线编程示例

Table 51 给出的编程示例定义以下配置。

- I/O 线 0 到 3 上 4 位输出端口 (应在单写操作中写入) , 开漏 , 有上拉电阻
- I/O 线 4 到 7 上 4 个输出信号 (例如驱动 LED) , 驱动高低电平 , 无上拉电阻
- I/O线8到11上4个输入信号(例如读取按钮操作状态) ,有上拉电阻、毛刺滤波器即输入变化中断
- I/O 线 12 到 15 上 4 个输入信号读取外部器件状态 (轮询 ,因此无输入变化中断) ,无上拉电阻 , 无毛刺滤波器
- I/O 线 16 到 19 分配给外设 A , 有上拉电阻
- I/O 线 20 到 23 分配给外设 B , 无上拉电阻
- I/O 线 24 到 27 分配给外设 A , 有输入变化中断 , 有上拉电阻

Table 51. 编程示例

寄存器	写入值
PIO_PER	0x0000 FFFF
PIO_PDR	0x0FFF 0000
PIO_OER	0x0000 00FF
PIO_ODR	0x0FFF FF00
PIO_IFER	0x0000 0F00
PIO_IFDR	0x0FFF F0FF
PIO_SODR	0x0000 0000
PIO_CODR	0x0FFF FFFF
PIO_IER	0x0F00 0F00
PIO_IDR	0x00FF F0FF
PIO_MDER	0x0000 000F
PIO_MDDR	0x0FFF FFF0
PIO_PUDR	0x00F0 00F0
PIO_PUER	0x0F0F FF0F
PIO_ASR	0x0F0F 0000
PIO_BSR	0x00F0 0000
PIO_OWER	0x0000 000F
PIO_OWDR	0x0FFF FFF0

并行输入 / 输出控制器 (PIO) 用户接口

PIO控制器控制的每个I/O线都与 PIO控制器用户接口寄存器中的某一位相关。每个寄存器有32位宽。若一个并行 I/O 线未定义，对相应位写入无效。未定义位读为零。若 I/O 线未与外设复用， I/O 线由 PIO 控制器控制且 PIO_PSR 返回 1。

Table 52. 并行输入 / 输出控制器 (PIO) 寄存器映射

偏移	寄存器	名称	访问类型	复位值
0x0000	PIO 使能寄存器	PIO_PER	只写	–
0x0004	PIO 禁用寄存器	PIO_PDR	只写	–
0x0008	PIO 状态寄存器 (1)	PIO_PSR	只读	0x0000 0000
0x000C	保留			
0x0010	输出使能寄存器	PIO_OER	只写	–
0x0014	输出禁用寄存器	PIO_ODR	只写	–
0x0018	输出状态寄存器	PIO_OSR	只读	0x0000 0000
0x001C	保留			
0x0020	毛刺输入滤波器使能寄存器	PIO_IFER	只写	–
0x0024	毛刺输入滤波器禁用寄存器	PIO_IFDR	只写	–
0x0028	毛刺输入滤波器状态寄存器	PIO_IFSR	只读	0x0000 0000
0x002C	保留			
0x0030	置位输出数据寄存器	PIO_SODR	只写	–
0x0034	清零输出数据寄存器	PIO_CODR	只写	–
0x0038	输出数据状态寄存器 (2)	PIO_ODSR	只读	0x0000 0000
0x003C	引脚数据状态寄存器 (3)	PIO_PDSR	只读	
0x0040	中断使能寄存器	PIO_IER	只写	–
0x0044	中断禁用寄存器	PIO_IDR	只写	–
0x0048	中断屏蔽寄存器	PIO_IMR	只读	0x00000000
0x004C	中断状态寄存器 (4)	PIO_ISR	只读	0x00000000
0x0050	多驱动使能寄存器	PIO_MDER	只写	–
0x0054	多驱动禁用寄存器	PIO_MDDR	只写	–
0x0058	多驱动状态寄存器	PIO_MDSR	只读	0x00000000
0x005C	保留			
0x0060	上拉禁用寄存器	PIO_PUDR	只写	–
0x0064	上拉使能寄存器	PIO_PUER	只写	–
0x0068	上拉状态寄存器	PIO_PUSR	只读	0x00000000
0x006C	保留			

Table 52. 并行输入 / 输出控制器 (PIO) 寄存器映射

偏移	寄存器	名称	访问类型	复位值
0x0070	外设 A 选择寄存器 (5)	PIO_ASR	只写	–
0x0074	外设 B 选择寄存器 (5)	PIO_BSR	只写	–
0x0078	AB 状态寄存器 (5)	PIO_ABSR	只读	0x00000000
0x007C - 0x009C	保留			
0x00A0	输出写使能	PIO_OWER	只写	–
0x00A4	输出写禁用	PIO_OWDR	只写	–
0x00A8	输出写状态寄存器	PIO_OWSR	只读	0x00000000
0x00AC - 0x00FC	保留			

Notes:
1. PIO_PSR 复位值取决于产品。
2. PIO_ODSR 根据 PIO_OWSR I/O 线不同为只读或读 / 写。
3. PIO_PDSR 复位值取决于 I/O 线电平。
4. PIO_ISR 复位值为 0x0。但当首次读该寄存器时可能会由于发生输入变化而导致读出一个不同的值。
5. 仅该系列寄存器可通过向第一个寄存器写 1 清除状态，对第二个寄存器写 1 置位状态。

PIO 控制器 PIO 使能寄存器

寄存器名称 : PIO_PER

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: PIO 使能**

0 = 无效。

1 = 使能 PIO 来控制相应引脚 (禁用引脚外设控制)。

PIO 控制器 PIO 禁用寄存器

寄存器名称 : PIO_PDR

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: PIO 禁用**

0 = 无效。

1 = 禁用 PIO 控制相应引脚 (使能引脚外设控制)。

PIO 控制器 PIO 状态寄存器

寄存器名称 : PIO_PSR

访问类型 : 只读

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: PIO 状态**

0 = 相应的 I/O 线上 PIO 无效 (外设激活)。

1 = 相应的 I/O 线上 PIO 有效 (外设无效)。

PIO 控制器输出使能寄存器

寄存器名称 : PIO_OER

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输出使能**

0 = 无效。

1 = 使能 I/O 线上输出。

PIO 控制器输出禁用寄存器

寄存器名称 : PIO_ODR

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输出禁用**

0 = 无效。

1 = 禁用 I/O 线上输出。

PIO 控制器输出状态寄存器

寄存器名称 : PIO_OSR

访问类型 : 只读

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输出状态**

0 = I/O 线为纯输入。

1 = I/O 线输出使能。

PIO 控制器输入滤波器使能寄存器

寄存器名称 : PIO_IFER

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输入滤波器使能**

0 = 无效。

1 = 使能 I/O 线上输入毛刺滤波器。

PIO 控制器输入滤波器禁用寄存器

寄存器名称 : PIO_IFDR

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输入滤波器禁用**

0 = 无效。

1 = 禁用 I/O 线上输入毛刺滤波器。

PIO 控制器输入滤波器状态寄存器

寄存器名称 : PIO_IFSR

访问类型 : 只读

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输入滤波器状态**

0 = I/O 线上输入毛刺滤波器禁用。

1 = I/O 线上输入毛刺滤波器使能。

PIO 控制器置位输出数据寄存器

寄存器名称 : PIO_SODR

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 置位输出数据**

0 = 无效。

1 = 设置在 I/O 线上驱动的数据。

PIO 控制器输出数据清零寄存器

寄存器名称 : PIO_CODR

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 设置输出数据**

0 = 无效。

1 = 清除在 I/O 线上驱动的数据。

PIO 控制器输出数据状态寄存器

寄存器名称 : PIO_ODSR

访问类型 : 只读或读 / 写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输出数据状态**

0 = 驱动到 I/O 线上的数据为 0。

1 = 驱动到 I/O 线上的数据为 1。

PIO 控制器引脚数据状态寄存器

寄存器名称 : PIO_PDSR

访问类型 : 只读

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输出数据状态**

0 = I/O 线电平为 0。

1 = I/O 线电平为 1。

PIO 控制器中断使能寄存器

寄存器名称 : PIO_IER

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输入变化中断使能**

0 = 无效。

1 = 使能 I/O 线上输入变化中断。

PIO 控制器中断禁用寄存器

寄存器名称 : PIO_IDR

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输入变化中断禁用**

0 = 无效。

1 = 禁用 I/O 线上输入变化中断。

PIO 控制器中断屏蔽寄存器

寄存器名称 : PIO_IMR

访问类型 : 只读

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输入变化中断屏蔽**

0 = I/O 线上输入变化中断禁用。

1 = I/O 线上输入变化中断使能。

PIO 控制器中断状态寄存器

寄存器名称 : PIO_ISR

访问类型 : 只读

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输入变化中断状态**

0 = 上次 PIO_ISR 读后或复位后， I/O 线上未检测到输入变化。

1 = 上次 PIO_ISR 读后或复位后， I/O 线上至少检测到一次输入变化。

PIO 多驱动使能寄存器

寄存器名称 : PIO_MDER

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 多驱动使能**

0 = 无效。

1 = 使能 I/O 线上多驱动。

PIO 多驱动禁用寄存器

寄存器名称 : PIO_MDDR

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 多驱动禁用**

0 = 无效。

1 = 禁用 I/O 线上多驱动。

PIO 多驱动状态寄存器

寄存器名称 : PIO_MDSR

访问类型 : 只读

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 多驱动状态**

0 = I/O 线上多驱动禁用。引脚驱动为高低电平。

1 = I/O 线上多驱动使能。引脚仅驱动为低电平。

PIO 上拉禁用寄存器

寄存器名称 : PIO_PUDR

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 上拉禁用**

0 = 无效。

1 = 禁用 I/O 线上拉电阻。

PIO 上拉使能寄存器

寄存器名称 : PIO_PUER

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 上拉使能**

0 = 无效。

1 = 使能 I/O 线上拉电阻。

PIO 上拉状态寄存器

寄存器名称 : PIO_PUSR

访问类型 : 只读

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 上拉状态**

0 = I/O 线上拉电阻使能。

1 = I/O 线上拉电阻禁用。

PIO 外设 A 选择寄存器

寄存器名称 : PIO_ASR

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 外设 A 选择**

0 = 无效。

1 = I/O 线分配给外设 A。

PIO 外设 B 选择寄存器

寄存器名称 : PIO_BSR

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 外设 B 选择**

0 = 无效。

1 = I/O 线分配给外设 B。

PIO 外设 A B 状态寄存器

寄存器名称 : PIO_ABSR

访问类型 : 只读

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 外设 A B 状态**

0 = I/O 线分配给外设 A。

1 = I/O 线分配给外设 B。

PIO 输出写使能寄存器

寄存器名称 : PIO_OWER

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输出写使能**

0 = 无效。

1 = 使能 I/O 线对 PIO_ODSR 写。

PIO 输出写禁用寄存器

寄存器名称 : PIO_OWDR

访问类型 : 只写

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输出写禁用**

0 = 无效。

1 = 禁用 I/O 线对 PIO_ODSR 写。

PIO 输出写状态寄存器

寄存器名称 : PIO_OWSR

访问类型 : 只读

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

- **P0-P31: 输出写状态**

0 = 写 PIO_ODSR 不影响 I/O 线。

1 = 写 PIO_ODSR 影响 I/O 线。

串行外设接口 (SPI)

概述

串行外设接口 (SPI) 电路是同步串行数据链接，在主机或从机模式下提供于外部器件的特性。若外部处理器与系统连接，它还使能处理器间通信。

串行外设接口实质上是一个将串行传输数据位发送到其它 SPI 的移位寄存器。数据传输时，一个 SPI 系统作为“主机”控制数据流，其它 SPI 作为“从机"，主机控制数据的移入与移出。不同的 CPU 可轮流作为主机 (多主机协议与单主机协议不同，单主机协议中只有一个 CPU 始终作为主机，其它 CPU 始终作为从机) 且一个主机可同时将数据移入多从机。但只允许单从机将其数据写入主机。

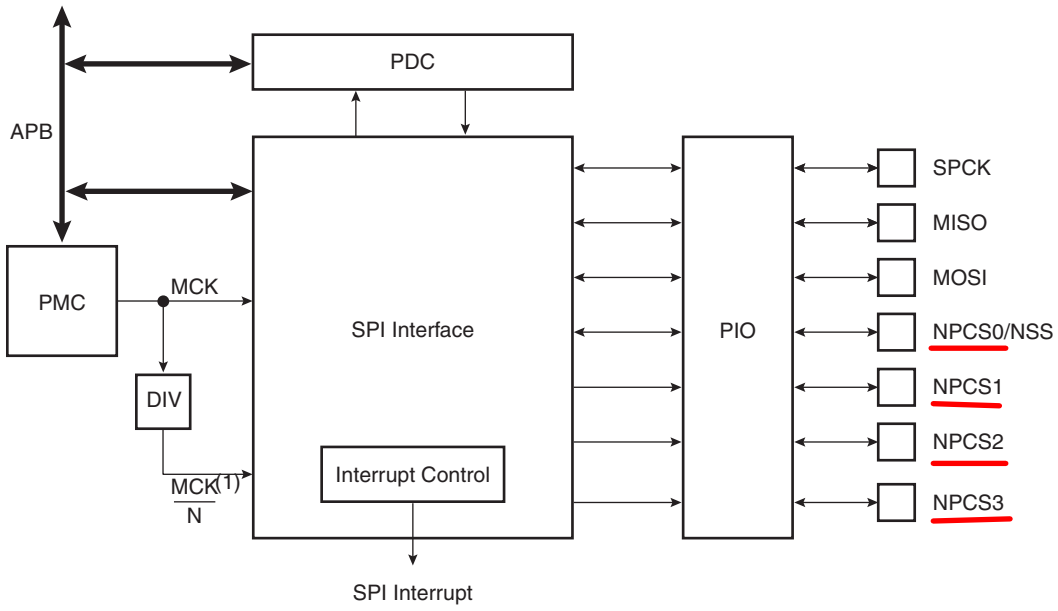
当主机插入 NSS 信号时，选定一个从机。若有多从机存在，主机对每个从机产生一个独立的从机选择信号 (NPCS)。

SPI 系统包括两条数据线及两条控制线：

- 主机输出从机输入 (MOSI)：该数据线将主机输出数据作为从机输入移入。
- 主机输入从机输出 (MISO)：该数据线将从机输出作为主机输入。传输时，只有单从机传输数据。
- 串行时钟 (SPCK)：该控制线由主机驱动，用来调节数据流。主机传输数据波特率可变；每传输一位，产生一个 SPCK 周期。
- 从机选择 (NSS)：该控制线允许硬件开关从机。

方框图

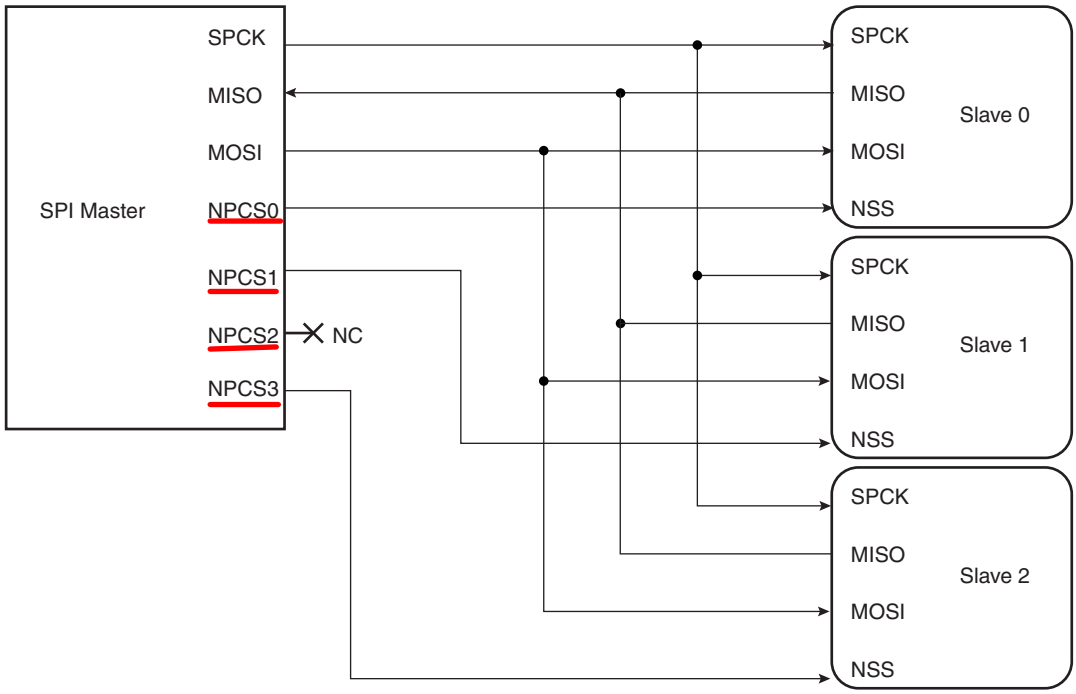
Figure 81. 方框图



Note: 1. N = 32

应用框图

Figure 82. 应用框图：单主机 / 多从机



信号说明

Table 53. 信号说明

引脚名称	引脚说明	类型	
		主机	从机
MISO	主入从出	输入	输出
MOSI	主出从入	输出	输入
SPCK	串行时钟	输出	输入
NPCS1-NPCS3	外设片选	输出	未用
NPCS0/NSS	外设片选 / 从机选择	输出	输入

附属产品

I/O 线

该引脚用来连接适用的外设，可与 PIO 线复用。程序员要先对 PIO 控制器编程，将 SPI 引脚分配给外设。

电源管理

SPI 可由电源管理控制器 (PMC) 提供时钟，因此，程序员必须先配置 PMC 以使能 SPI 时钟。

中断

SPI接口有与高级中断控制器(AIC)连接的中断线。处理SPI中断请求须在配置SPI前对AIC编程。

功能描述

工作模式

SPI 工作在主机模式或从机模式下。

通过将模式寄存器 MSTR 位写 1 将 SPI 工作在主机模式下。引脚 NPCS0 到 NPCS3 配置为输出，SPCK 引脚被驱动，MISO 线与接收器输入连接且 发送器驱动 MOSI 作为输出。

若 MSTR 位写入 0，SPI 工作在从机模式下。MISO 线由发送器输出驱动，MOSI 线与接收器输入连接，发送器驱动 SPCK 引脚以实现与接收器同步。NPCS0 引脚变为输入，并作为从机选择信号 (NSS) 使用。引脚 NPCS1 到 NPCS3 未驱动，可用于其它功能。

两种工作模式下数据传输同样可编程。只有在主机模式下激活波特率发生器。

数据传输

数据传输有四种极性与相位。时钟极性由片选寄存器 CPOL 位编程得到。时钟相位由 NCPHA 位编程得到。这两个参数确定数据在哪个时钟边沿驱动与采样。每个参数有两种状态，组合后有四种可能。因此，一对主机 / 从机必须使用相同的参数对值来进行通信。若使用多从机，且固定为不同的配置，主机与不同从机通信时必须重新配置。

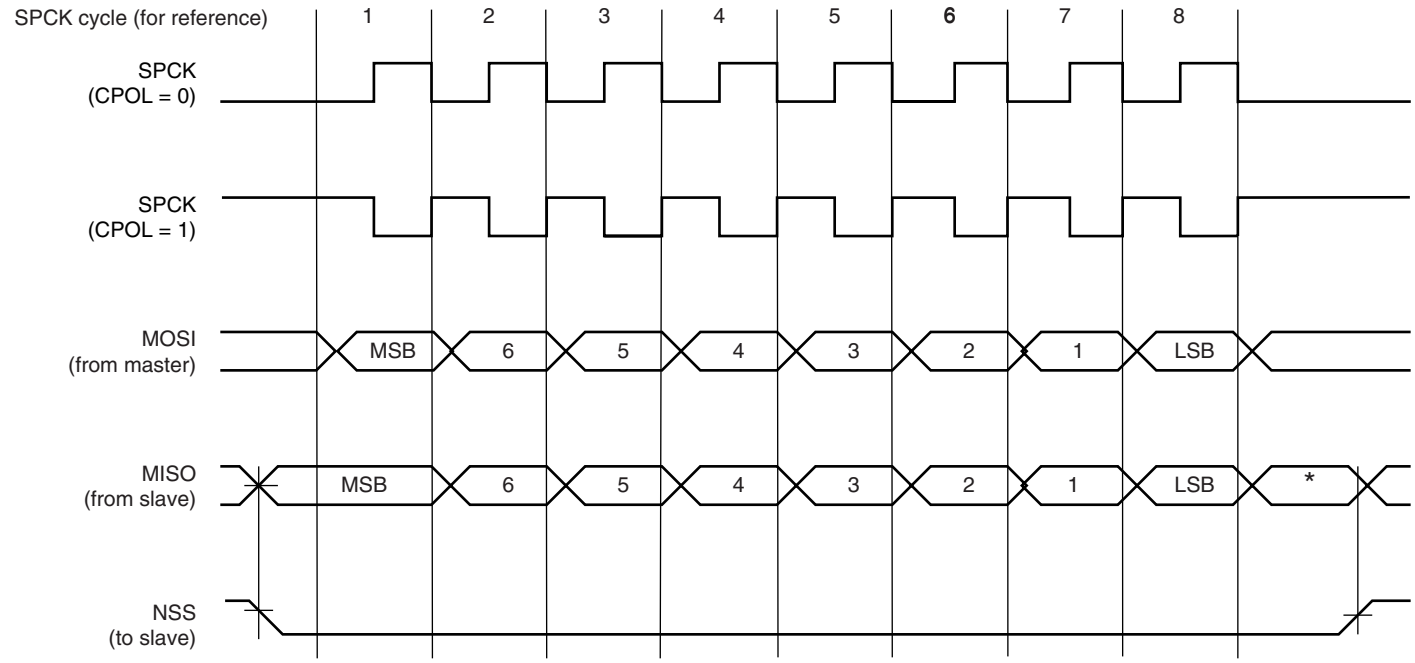
Table 54 给出四种模式与相应的参数设置。

Table 54. SPI 总线协议模式

SPI 模式	CPOL	CPHA
0	0	1
1	0	0
2	1	1
3	1	0

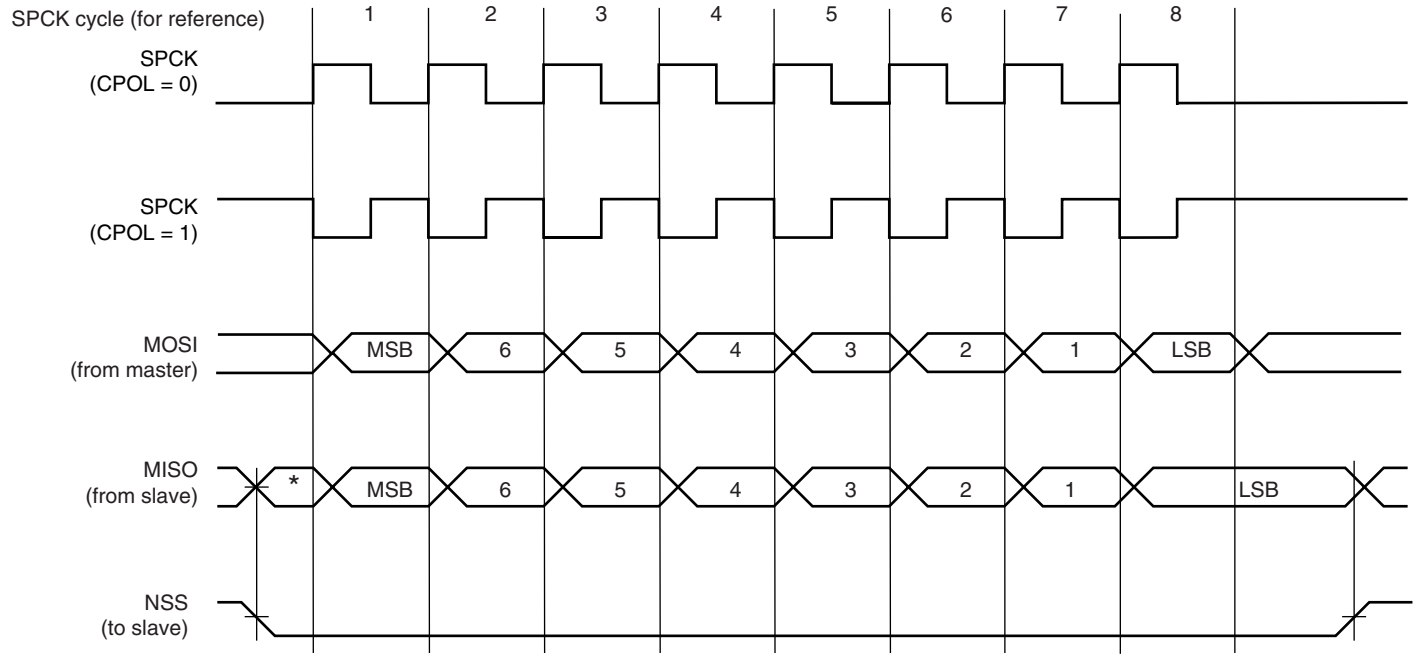
Figure 83 与 Figure 84 给出数据传输示例。

Figure 83. SPI 传输格式 (NCPHA = 1，每次传输 8 位)



* Not defined, but normally MSB of previous character received.

Figure 84. SPI 传输格式 (NCPHA = 0，每次传输 8 位)



* Not defined but normally LSB of previous character transmitted.

主机工作模式

当配置为主机模式， SPI 工作时钟由内部可编程波特率发生器产生。它完全控制与 SPI 总线连接的从机数据传输。 SPI 驱动片选线为从机及串行时钟信号 (SPCK)。

SPI 有两个保持寄存器 ,发送数据寄存器与接收数据寄存器及单移位寄存器。保持寄存器将数据流保持在一个恒定的速率上。

使能 SPI 后，当处理器写入 SPI_TDR(发送数据寄存器) 时，数据开始传输。被写数据立即发往移位寄存器并开始在 SPI 总线上传输。当移位寄存器中数据移到 MOSI 线上时，对 MISO 线采样并移入移位寄存器。没有接收，发送不能出现。

若PCS域未选择从机 ,当写入SPI_TDR时不会启动传输。PCS域通过在可变模式下写SPI_TDR来置位，或在固定模式下写 SPI_MR 来置位，由 PCS 域值决定。

若在传输时有新数据写入 SPI_TDR，它将保持当前值直到传输完成。然后接收到的数据由移位寄存器送到 SPI_RDR 中， SPI_TDR 中数据载入移位寄存器并启动新的传输。

将写在 SPI_TDR 中的数据送往移位寄存器由状态寄存器 (SPI_SR) 的 TDRE 位 (发送数据寄存器空) 表示。当新数据写入 SPI_TDR 时，该位清零。 TDRE 位用来触发发送 PDC 通道。

传输结束由 SPI_SR 寄存器中的 TXEMPTY 标志表示。若最后传输的传输延迟 (DLYBCT) 大于 0， TXEMPTY 在上述延迟完成后置位。此时主机时钟 (MCK) 可关闭。

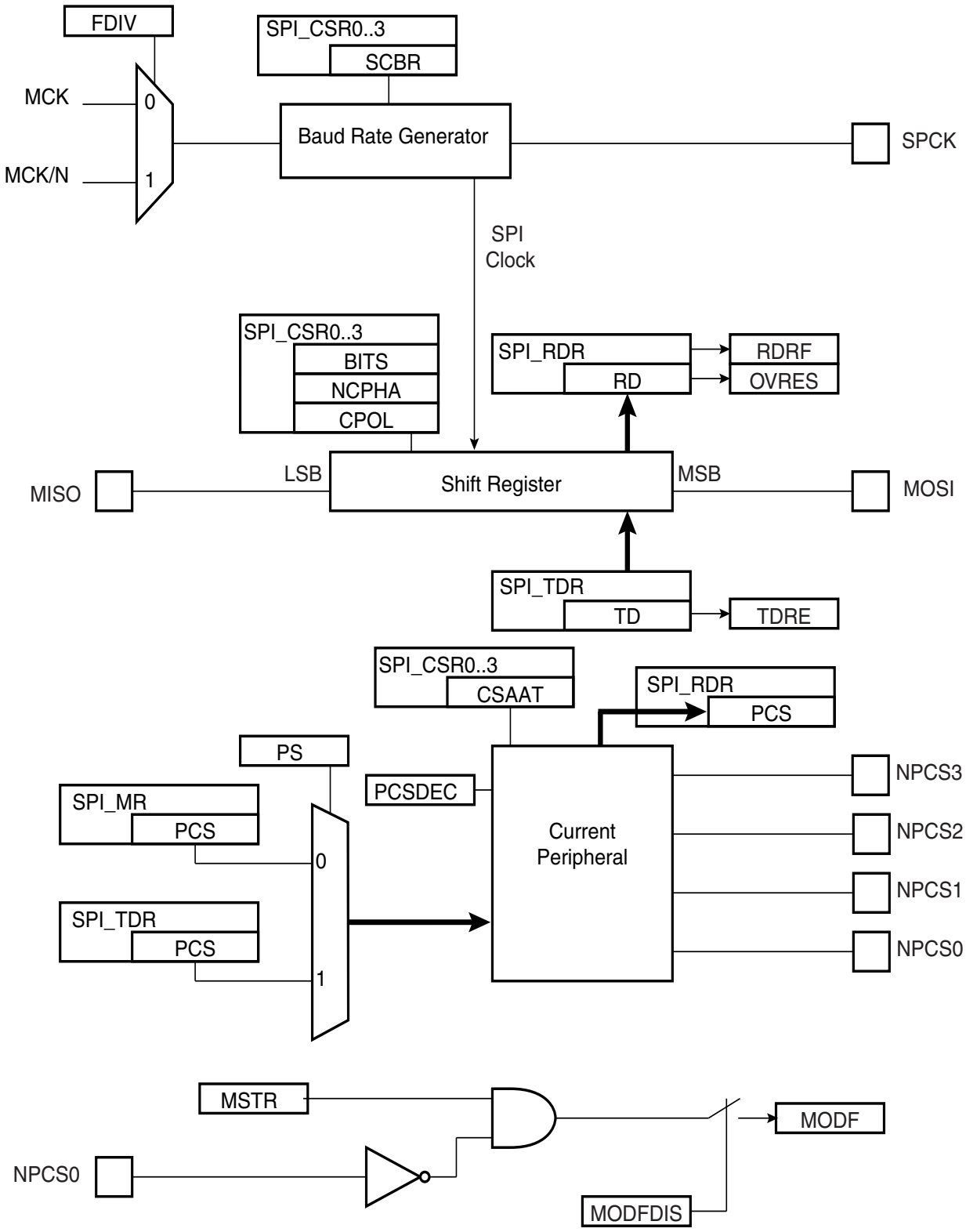
接收 SPI_RDR 中来自移位寄存器的数据由 SPI_SR 寄存器的 RDRF 位 (接收数据寄存器满) 来表示。当读取接收数据时， RDRF 位清零。

若在接收新数据前 SPI_RDR (接收数据寄存器) 仍未被读取 , SPI_SR 中溢出错误位 (OVRES) 置位。该标志置位时 ,SPI_RDR 中不会载入数据。用户必须读状态寄存器以对 OVRES 位清零。

Figure 85 on page 233 给出主机模式下SPI框图。Figure 86 on page 234给出传输处理流程图。

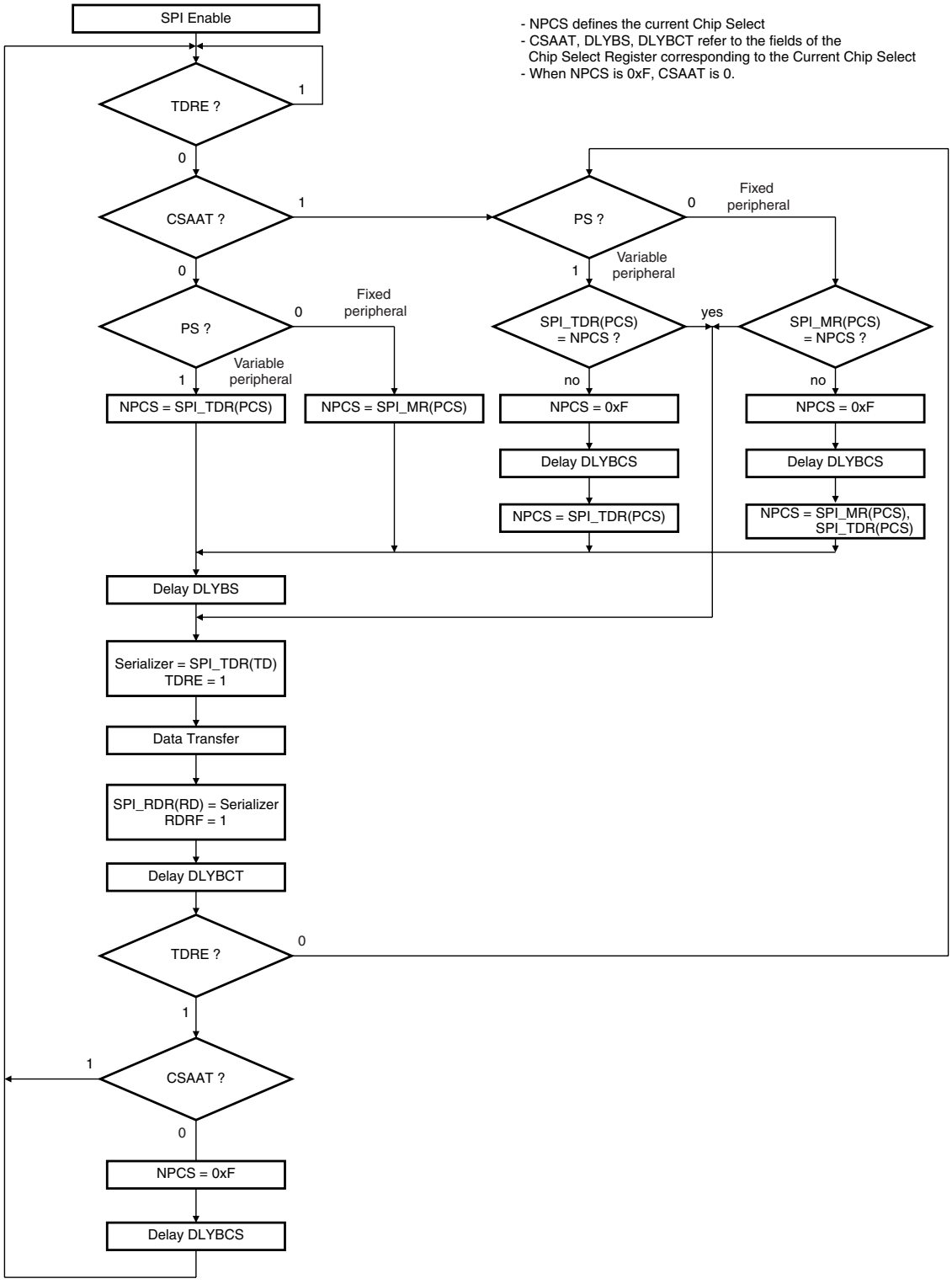
主机模式方框图

Figure 85. 主机模式方框图



主机模式流程图

Figure 86. 主机模式流程图



时钟产生

SPI波特率时钟由主机时钟(MCK)分频或主机时钟32与2或255乘积值分频。选择主机时钟还是主机时钟 N 分频由模式寄存器中 FDIV 值决定。

这使允许的最大工作波特率达到 1/2 主机时钟，最小值为 MCK 除以 255*32。

禁止对 SCBR 域编程为 0。当 SCBR 为 0 时触发传输结果未知。

复位后 SCBR 为 0，因此在首次传输前用户必须将其设定为一个有效值。

每个片选可独立对分频器定义，必须在片选寄存器的 SCBR 域编程。这允许 SPI 对每个外设接口自动调整波特率而不需重新编程。

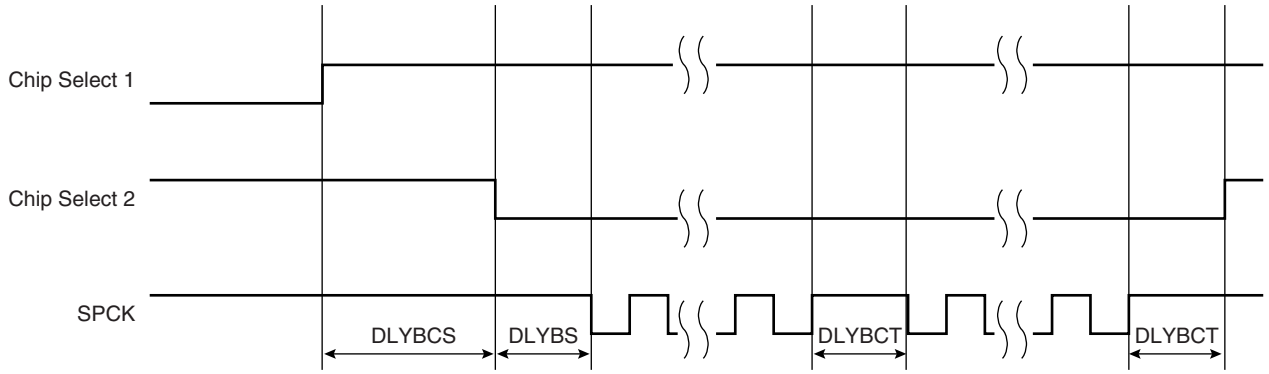
传输延迟

Figure 87 给出片选传输改变及在在相同芯片上连续传输。有三种延迟可编程以修改传输波形：

- 片选间延迟，对于所有片选只可通过写模式寄存器的 DLYBCS 域改变一次。允许在释放芯片及开始新传输前插入一个延时。
- SPCK 前延迟，通过写 DLYBS 域实现，对每个片选独立可编程。允许在片选出现后 SPCK 启动延迟。
- 连续传输延迟，通过写 DLYBCT 域实现，对每个片选独立可编程。在同一芯片两传输间插入一个延迟。

这些延迟允许 SPI 调整与外设连接及它们的速度及总线释放时间。

Figure 87. 可编程延迟



外设选择

通过 NPCS0 到 NPCS3 信号选择串行外设。默认情况下，传输前后 NPCS 信号为高。

有两种方式执行外设选择：

- 固定外设选择：SPI 只与一个外设交换数据
- 可变外设选择：SPI 与多个外设交换数据

固定外设选择通过在 SPI_MR(模式寄存器) 的 PS 位写零激活。此时，当前外设由 SPI_MR 的 PCS 域定义且片选寄存器的 PCS 域无效。

可变外设选择通过 PS 位写一激活。当前外设由 SPI_MR 的 PCS 域选择。即每个新数据可定义外设选择。

固定外设选择允许单外设缓冲器传输。使用 PDC 是最佳方式，因为存储器与 SPI 数据传输均为 8 或 16 位。但是，改变外设选择需要将模式寄存器重新编程。

可变外设选择允许多外设缓冲器传输而不需对模式寄存器重新编程。写入 SPI_TDR 的数据为 32 位宽并定义将实数据发送到外设。该模式下使用的 PDC 为 32 位宽缓冲器，数据低位在先，PCS与LASTXFER域高位在先，但SPI仍通过片选配置寄存器的MISO与MOSI线控制传输位数(8 到 16)。这对于存储器缓冲器并非最佳方式，但它是几个外设无处理器干扰下最有效的数据交换方式。

外设片选译码

用户可通过对片选线 NPCS0 到 NPCS3 解码，实现 SPI 对高达 15 个外设的操作。通过在模式寄存器 PCSDEC 位写 1 使能。

当不译码操作时，SPI 保证只激活一个片选，即每次拉低。若 PCS 域中两位为低，只将最低序号的片选拉低。

当译码操作时，SPI 直接输出由模式寄存器或发送数据寄存器定义的 PCS 域值 (由 PS 决定)。

由于 SPI 默认值为 0xF(即所有片选线为 1)，当没有处理传输时，只可对 15 个外设译码。

SPI 只有 4 个片选寄存器，而非 15 个。因此，当译码激活时，每个片选定义 4 个外设特性。例如，SPI_CRS0 定义外部译码外设 0 到 3 特性，对应于 PCS 值 0x0 到 0x3。因此，用户必须确保译码片选线 0 到 3、 4 到 7、 8 到 11 及 12 到 14 连接的外设兼容。

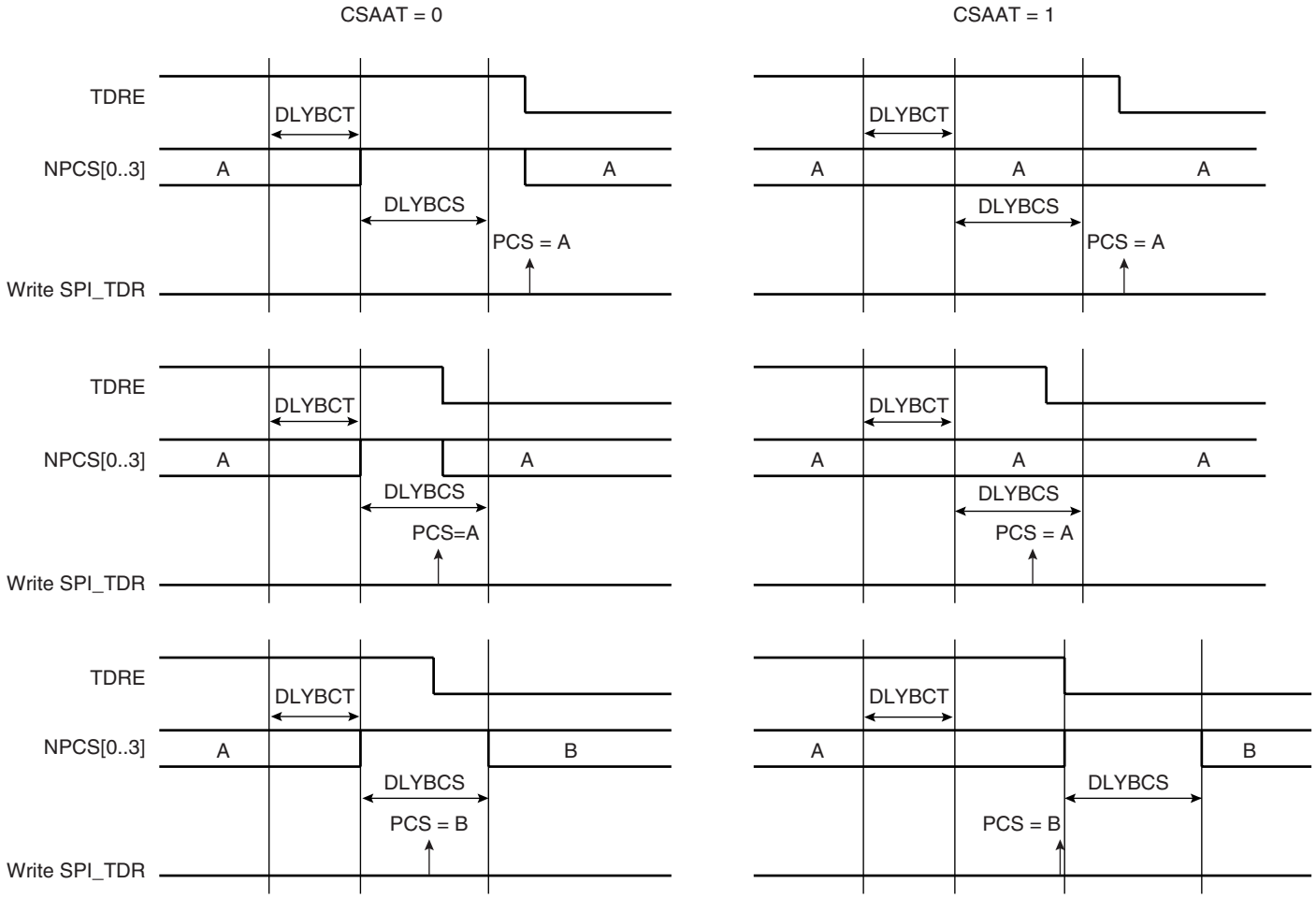
外设选择取消

当正常工作时，传输的最后数据写入 SPI_TDR 后，所有 NPCS 线拉高。若处理器响应中断时间太长可能会引起运行时间错误，并可能导致与某些需要传输过程中片选线保持激活的串行外设连接困难。

为方便与此类器件的连接，片选寄存器 CSAAT(传输后片选激活) 为 1。这允许片选线保持其当前状态 (低 = 激活) 直到出现其它外设传输请求。

Figure 88 给出不同的外设选择取消情况及 CSAAT 位影响。

Figure 88. 外设选择取消



模式错误检测

当 SPI 编程为主机模式且外部主机将 NPCS0/NSS 信号驱动为低电平时，检测到模式错误。由于该引脚通常配置为开漏的，所以应在 NPCS0 线上连接一个上拉电阻，以确保高电平且不会检测到伪模式错误。

当检测到模式错误， MODF 位在读 SPI_SR 前置位，而 SPI 自动禁用直到在 SPI_CR(控制寄存器) 的 SPIEN 位写 1 将其重新使能为止。

默认情况下，模式错误检测电路使能。用户可通过设置 SPI_MR 中的 MODFDIS 位禁用模式错误检测。

SPI 从机模式

从机模式下， SPI 处理器数据位时钟由 SPI 时钟引脚 (SPCK) 提供。

SPI在由外部主机接收串行时钟前等待NSS激活。当NSS下降，时钟在串行器上生效，处理片选寄存器 0 (SPI_CSR0)BITS 域定义的位数 。这些位处理的相位与极性由 SPI_CSR0 的 NCPHA 与 CPOL 位定义。注意，当 SPI 编程为从机模式时，其它片选寄存器的 BITS、CPOL 及 NCPHA 位无效。

位移出到 MISO 线并在 MOSI 线上采样。

当所有位处理时，接收到的数据发送到接收数据寄存器， RDRF 位升高。若数据传输时 RDRF 已经为高，溢出位拉高并终止向 SPI_RDR 的数据传输。

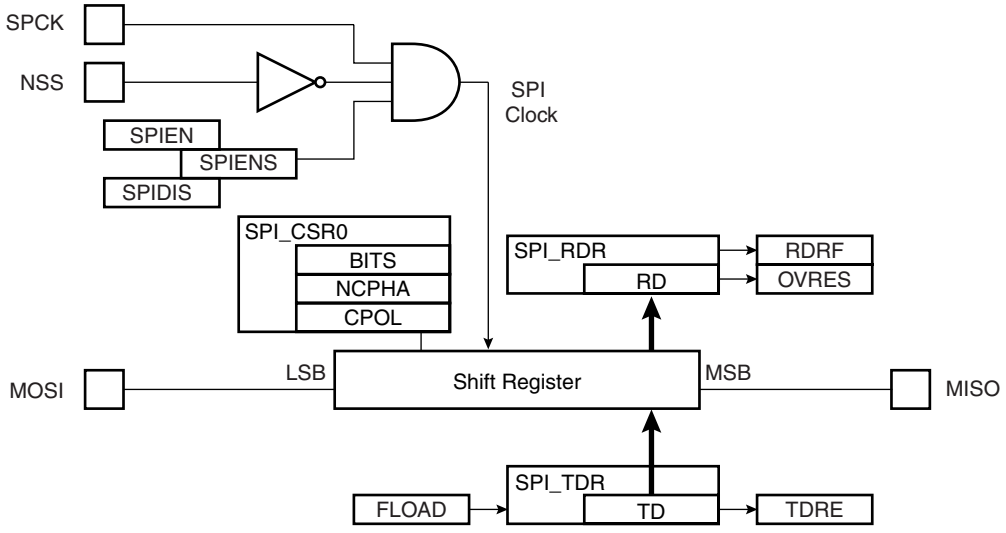
当传输启动，数据由移位寄存器移出。若发送数据寄存器 (SPI_TDR) 中没有数据写入，则发送最后收到的数据。若在上次复位后未收到数据，发送的所有位为低，因为移位寄存器复位为 0。

当首数据写入 SPI_TDR，立即向移位寄存器传输并将 TDRE 位拉高。若新数据已写入，将保存在 SPI_TDR 中直到传输出现，即 NSS 下降且 SPCK 引脚上出现有效时钟。当传输出现，最后写入 SPI_TDR 的数据传入移位寄存器并将 TDRE 位拉高。这使能单传输临界变化频繁更新。

然后，新数据由发送数据寄存器载入移位寄存器中。若没有发送字符，即自从上次由 SPI_TDR 载入移位寄存器后，没有字符写入 SPI_TDR，移位寄存器不变并重新发送最后收到的字符。

Figure 89 给出从机模式下 SPI 框图。

Figure 89. 从机模式功能框图



串行外设接口 (SPI) 用户接口

Table 55. 串行外设接口 (SPI) 寄存器映射

偏移	寄存器	寄存器名称	访问类型	复位值
0x00	控制寄存器	SPI_CR	只写	---
0x04	模式寄存器	SPI_MR	读 / 写	0x0
0x08	接收数据寄存器	SPI_RDR	只读	0x0
0x0C	发送数据寄存器	SPI_TDR	只写	---
0x10	状态寄存器	SPI_SR	只读	0x000000F0
0x14	中断使能寄存器	SPI_IER	只写	---
0x18	中断禁用寄存器	SPI_IDR	只写	---
0x1C	中断屏蔽寄存器	SPI_IMR	只读	0x0
0x20 - 0x2C	保留			
0x30	片选寄存器 0	SPI_CSR0	读 / 写	0x0
0x34	片选寄存器 1	SPI_CSR1	读 / 写	0x0
0x38	片选寄存器 2	SPI_CSR2	读 / 写	0x0
0x3C	片选寄存器 3	SPI_CSR3	读 / 写	0x0
0x004C - 0x00FC	保留	–	–	–
0x100 - 0x124	保留给 PDC			

SPI 控制寄存器

寄存器名称 : SPI_CR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	LASTXFER
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
SWRST	–	–	–	–	–	SPIDIS	SPIEN

- **SPIEN: SPI 使能**

0 = 无效。

1 = 使能 SPI 发送与接收数据。

- **SPIDIS: SPI 禁用**

0 = 无效。

1 = 禁用 SPI。

所有引脚设置为输入模式，无数据发送或接收。

若正处理传输，传输结束后禁用 SPI。

当控制寄存器写入时若 SPIEN 与 SPIDIS 均为 1， SPI 禁用。

- **SWRST: SPI 软件复位**

0 = 无效。

1 = 复位 SPI。执行 SPI 接口的软件触发硬件复位。

- **LASTXFER: 最后传输**

0 = 无效。

1 = 在写入TD的字符传输后，当前的NPCS将失效。当CSAAT置位，在TD传输完成后通过将相应的NPCS线拉高关闭与当前串行外设通信。

SPI 模式寄存器

寄存器名称 : SPI_MR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
DLYBCS							
23	22	21	20	19	18	17	16
–	–	–	–	PCS			
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
LLB	–	–	MODFDIS	FDIV	PCSDEC	PS	MSTR

- **MSTR: 主机 / 从机模式**

0 = SPI 为从机模式。

1 = SPI 为主机模式。

- **PS: 外设选择**

0 = 固定外设选择。

1 = 可变外设选择。

- **PCSDEC: 片选译码**

0 = 片选直接与外设连接

1 = 四个片选与 4 到 16 位译码器连接。

当 PCSDEC 等于 1，使用外部 4 到 16 位译码器可产生 15 个片选信号。片选寄存器通过下列规则定义 16 个片选特性：

SPI_CSR0 定义外设片选信号 0 到 3。

SPI_CSR1 定义外设片选信号 4 到 7。

SPI_CSR2 定义外设片选信号 8 到 11。

SPI_CSR3 定义外设片选信号 12 到 15。

- **FDIV: 时钟选择**

0 = SPI 时钟为 MCK。

1 = SPI 时钟为 MCK/N。

- **MODFDIS: 模式错误检测**

0 = 模式错误检测使能。

1 = 模式错误检测禁用。

- **LLB: 本地回环使能**

0 = 本地回环路径禁用。

1 = 本地回环路径使能。

LLB 在主机模式下控制数据串行器的本地回环。

- **PCS: 外设片选**

该域仅适用于固定外设选择有效 (PS = 0)。

若 PCSDEC = 0：

PCS = xxx0 NPCS[3:0] = 1110
PCS = xx01 NPCS[3:0] = 1101
PCS = x011 NPCS[3:0] = 1011
PCS = 0111 NPCS[3:0] = 0111
PCS = 1111 禁用 (未选择外设)
(x = 不必在意)

若 PCSDEC = 1：

NPCS[3:0] 输出信号 = PCS。

- **DLYBCS: 片选延迟**

该域定义 NPCS 无效到其它 NPCS 有效间的延迟。DLYBCS 时间保证片选不重叠并解决外设长时间数据流引起的总线竞争。

若 DLYBCS 小于或等于 6， 6 个 MCK 周期 (若 FDIV 置位， 6*N MCK 周期) 将缺省插入。

其它情况下，按照下列等式确定延迟：

若 FDIV 为 0：

$$\text{Delay Between Chip Selects} = \frac{DLYBCS}{MCK}$$

若 FDIV 为 1：

$$\text{Delay Between Chip Selects} = \frac{DLYBCS \times N}{MCK}$$

SPI 接收数据寄存器

寄存器名称 : SPI_RDR

访问类型 : 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	PCS			
15	14	13	12	11	10	9	8
RD							
7	6	5	4	3	2	1	0
RD							

- **RD: 接收数据**

SPI 接口收到的数据存于该寄存器的相应位中。未用位值为 0。

- **PCS: 外设片选**

仅在主机模式下，这些位表示传输结束后 NPCS 引脚值；其它情况下，这些位值为 0。

SPI 发送数据寄存器

寄存器名称 : SPI_TDR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	LASTXFER

23	22	21	20	19	18	17	16
–	–	–	–	PCS			

15	14	13	12	11	10	9	8
TD							

7	6	5	4	3	2	1	0
TD							

- **TD: 发送数据**

SPI 接口发送数据存于该寄存器中。要发送的信息必须以正确的格式写入数据发送寄存器中。

PCS: 外设片选

该域仅适用于可变外设选择有效 (PS = 1)。

若 PCSDEC = 0：

PCS = xxx0 NPCS[3:0] = 1110

PCS = xx01 NPCS[3:0] = 1101

PCS = x011 NPCS[3:0] = 1011

PCS = 0111 NPCS[3:0] = 0111

PCS = 1111 禁用 (未选择外设)

(x = 不必在意)

若 PCSDEC = 1：

NPCS[3:0] 输出信号 = PCS

- **LASTXFER: 最后传输**

0 = 无效。

1 = 在写入TD的字符传输后 , 当前的NPCS将失效。当CSAAT置位 , 在TD传输完成后通过将相应的NPCS线拉高关闭与当前串行外设通信。

SPI 状态寄存器

寄存器名称 : SPI_SR

访问类型 : 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	SPIENS
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: 接收数据寄存器满**

0 = 上次读 SPI_RDR 后未收到数据。

1 = 上次读 SPI_RDR 后已收到数据并由串行器发送到 SPI_RDR。

- **TDRE: 发送数据寄存器空**

0 = 数据已写入 SPI_TDR 但仍未传输到串行器。

1 = 最后写入发送数据寄存器的数据已传输到串行器。

当 SPI 禁用或复位时，TDRE 等于 0 。 SPI 使能命令设置该位为 1。

- **MODF: 模式错误检测**

0 = 上次读 SPI_SR 后未检测到模式错误。

1 = 上次读 SPI_SR 后出现模式错误。

- **OVRES: 溢出错误状态**

0 = 上次读 SPI_SR 后未检测到溢出错误。

1 = 上次读 SPI_SR 后出现溢出错误。

上次读 SPI_RDR 后当 SPI_RDR 至少两次载入串行器，出现溢出错误。

- **ENDRX: RX 缓冲结束**

0 = 上次写 SPI_RCR 或 SPI_RNCR 后接收计数寄存器仍未达到 0。

1 = 上次写 SPI_RCR 或 SPI_RNCR 后接收计数寄存器已达到 0。

- **ENDTX: TX 缓冲结束**

0 = 上次写 SPI_TCR 或 SPI_TNCR 后发送计数寄存器仍未达到 0。

1 = 上次写 SPI_TCR 或 SPI_TNCR 后发送计数寄存器已达到 0。

- **RXBUFF: RX 缓冲器满**

0 = SPI_RCR 或 SPI_RNCR 是非 0 值。

1 = SPI_RCR 与 SPI_RNCR 值为 0。

- **TXBUFE: TX 缓冲器空**

0 = SPI_TCR 或 SPI_TNCR 是非 0 值。

1 = SPI_TCR 与 SPI_TNCR 值为 0。

- **NSSR: NSS 上升**

0 = 上次读后未检测到 NSS 引脚上升沿。

1 = 上次读后 NSS 引脚出现上升沿。

- **TXEMPTY: 发送寄存器空**

0 = 数据立即写入 SPI_TDR。
1 = SPI_TDR 与内部移位器空。若已定义传输延迟，TXEMPTY 在延迟结束后置位。

- **SPIENS: SPI 使能状态**

0 = SPI 禁用。
1 = SPI 使能。

SPI 中断使能寄存器

寄存器名称： SPI_IER

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: 接收数据寄存器满中断使能**
- **TDRE: SPI 发送数据寄存器空中断使能**
- **MODF: 模式错误检测中断使能**
- **OVRES: 溢出错误中断使能**
- **ENDRX: 接收缓冲器结束中断使能**
- **ENDTX: 发送缓冲器结束中断使能**
- **RXBUFF: 接收缓冲器满中断使能**
- **TXBUFE: 发送缓冲器满中断使能**
- **TXEMPTY: 发送缓冲器空中断使能**
- **NSSR: NSS 上升中断使能**

0 = 无效。

1 = 使能相应中断。

SPI 中断禁用寄存器

寄存器名称 : SPI_IDR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: 接收数据寄存器满中断禁用**
- **TDRE: SPI 发送数据寄存器空中断禁用**
- **MODF: 模式错误检测中断禁用**
- **OVRES: 溢出错误中断禁用**
- **ENDRX: 接收缓冲器结束中断禁用**
- **ENDTX: 发送缓冲器结束中断禁用**
- **RXBUFF: 接收缓冲器满中断禁用**
- **TXBUFE: 发送缓冲器满中断禁用**
- **TXEMPTY: 发送缓冲器空中断禁用**
- **NSSR: NSS 上升中断禁用**

0 = 无效。

1 = 禁用相应中断。

SPI 中断屏蔽寄存器

寄存器名称 : SPI_IMR

访问类型 : 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	TXEMPTY	NSSR
7	6	5	4	3	2	1	0
TXBUFE	RXBUFF	ENDTX	ENDRX	OVRES	MODF	TDRE	RDRF

- **RDRF: 接收数据寄存器满中断屏蔽**
- **TDRE: SPI 发送数据寄存器空中断屏蔽**
- **MODF: 模式错误检测中断屏蔽**
- **OVRES: 溢出错误中断屏蔽**
- **ENDRX: 接收缓冲器结束中断屏蔽**
- **ENDTX: 发送缓冲器结束中断屏蔽**
- **RXBUFF: 接收缓冲器满中断屏蔽**
- **TXBUFE: 发送缓冲器满中断屏蔽**
- **TXEMPTY: 发送缓冲器空中断屏蔽**
- **NSSR: NSS 上升中断屏蔽**

0 = 相应中断未使能。

1 = 相应中断使能。

SPI 片选寄存器

寄存器名称： SPI_CSR0... SPI_CSR3

访问类型： 读 / 写

31	30	29	28	27	26	25	24
DLYBCT							
23	22	21	20	19	18	17	16
DLYBS							
15	14	13	12	11	10	9	8
SCBR							
7	6	5	4	3	2	1	0
BITS				CSAAT	–	NCPHA	CPOL

- **CPOL: 时钟极性**

0 = SPCK 无效状态值为逻辑 0。

1 = SPCK 无效状态值为逻辑 1。

CPOL 用来确定串行时钟 (SPCK) 的无效状态值，它和 NCPHA 一起使用产生主机与从机间所需的时钟 / 数据关系。

- **NCPHA: 时钟相位**

0 = 数据在 SPCK 起始边沿改变，在 SPCK 下一个边沿捕获。

1 = 数据在 SPCK 起始边沿捕获，在 SPCK 下一个边沿改变。

NCPHA 确定 SPCK 的哪个边沿引起数据改变，哪个边沿引起数据捕获。NCPHA 与 CPOL 一起使用产生主机与从机间所需的时钟 / 数据关系。

- **CSAAT: 传输后片选激活**

0 = 最后传输实现后外设片选上升。

1 = 最后传输实现后外设片选未上升。它保持激活直到不同的片选新传输请求。

- **BITS: 传输位**

BITS 域确定传输数据位数，不使用保留值。

BITS	传输位
0000	8
0001	9
0010	10
0011	11
0100	12
0101	13
0110	14
0111	15
1000	16
1001	保留
1010	保留
1011	保留
1100	保留
1101	保留
1110	保留
1111	保留

- **SCBR: 串行时钟波特率**

主机模式下，SPI 接口使用模块接收器由主机时钟 MCK 中导出 SPCK 波特率。可在 SCBR 域中写入 1 到 255 间的值来选择波特率。下列等式确定 SPCK 波特率：

若 FDIV 为 0：

$$\text{SPCK Baudrate} = \frac{MCK}{SCBR}$$

若 FDIV 为 1：

$$\text{SPCK Baudrate} = \frac{MCK}{(N \times SCBR)}$$

Note: N = 32

禁止将 SCBR 域编程为 0。当 SCBR 为 0 时触发传输将引起无法预期的结果。

复位时，SCBR 为 0 且用户在执行第一次传输前将 SCBR 编程为有效值。

- **DLYBS: SPCK 前延时**

该域定义由 NPCS 有效到第一次有效 SPCK 传输间的延迟。

当 DLYBS 等于 0，NPCS 有效到 SPCK 传输为 1/2 个 SPCK 时钟周期。

其它情况下，由下列等式决定延迟。

若 FDIV 为 0：

$$\text{Delay Before SPCK} = \frac{DLYBS}{MCK}$$

若 FDIV 为 1：

$$\text{Delay Before SPCK} = \frac{N \times DLYBS}{MCK}$$

Note: N = 32

- **DLYBCT: 连续传输间延迟**

该域定义无片选删除时相同外设的两次传输间的延迟。若需要的话，该延迟在每次传输后删除片选前插入。

当 DLYBCT 等于 0，连续传输中不插入延迟，且时钟保持其占空覆盖字符传输。

其它情况下，由下列等式决定延迟。

若 FDIV 为 0：

$$\text{Delay Between Consecutive Transfers} = \frac{32 \times DLYBCT}{MCK} + \frac{SCBR}{2MCK}$$

若 FDIV 为 1：

$$\text{Delay Between Consecutive Transfers} = \frac{32 \times N \times DLYBCT}{MCK} + \frac{N \times SCBR}{2MCK}$$

Note: N = 32

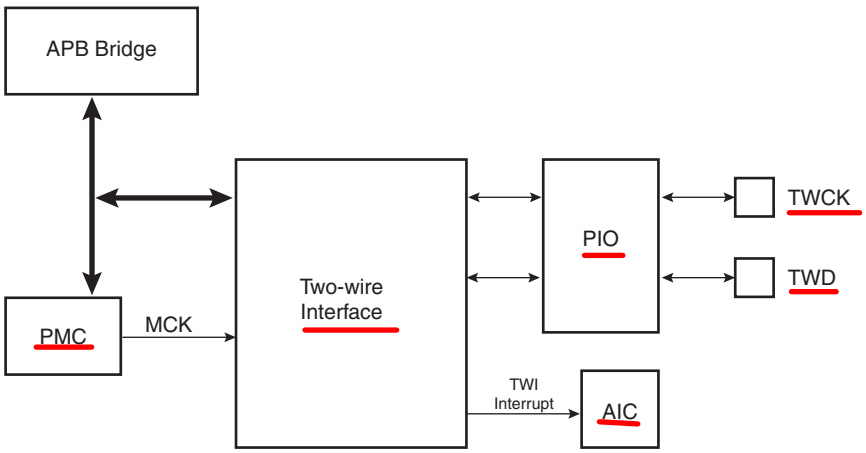
两线接口 (TWI)

概述

两线接口 (TWI) 由一根时钟线及一根传输速度达到 400 Kb/s 的数据线组成，以字节为单位进行传输。它适用于任何的 Atmel 两线总线串行 EEPROM 中。 TWI 可编程作为主机进行连续或单字节访问。可配置波特率发生器允许输出数据速率在内核时钟频率的一个宽范围内进行调整。

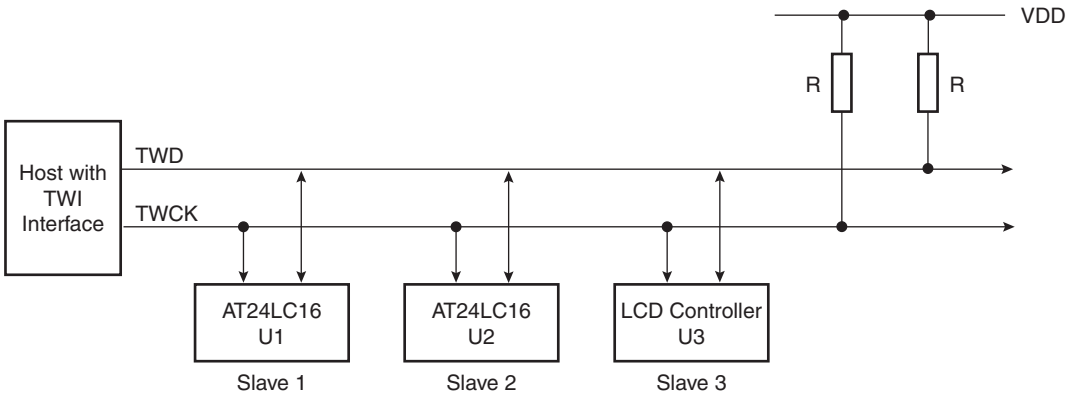
方框图

Figure 90. 方框图



应用框图

Figure 91. 应用框图



附属产品

I/O 线说明

Table 56. I/O 线说明

引脚名称	引脚说明	类型
TWD	两线串行数据	输入 / 输出
TWCK	两线串行时钟	输入 / 输出

TWD 与 TWCK 为双向线，通过当前源正极或上拉电阻连接 (见 Figure 91 on page 255)。当总线空闲时，两线均为高。连接到总线上的输出流水线必须有一个开漏或开集来执行线与功能。

TWD 与 TWCK 引脚可与 PIO 线复用。为使能 TWI，必须执行下列步骤：

- 将 PIO 控制器编程为：
  - 将 TWD 与 TWCK 指定为外设线。
  - 将 TWD 与 TWCK 定义为开漏。

电源管理

- 使能外设时钟。

TWI 接口可通过电源管理控制器 (PMC) 提供时钟，因此必须先配置 PMC 以使能 TWI 时钟。

中断

TWI接口有一条与高级中断控制器(AIC)连接的中断线。为处理中断，在配置TWI前必须对AIC编程。

功能说明

传输格式

TWD线上数据必须为8位。数据传输是高位在先；每字节后必须有应答信号。每次传输的字节数目没有限制 (见 Figure 93 on page 257)。

每次传输以 START 状态开始，以 STOP 状态停止 (见 Figure 92 on page 257)。

- 当 TWCK 为高时 TWD 由高变低定义为 START 状态。
- 当 TWCK 为高时 TWD 由低变高定义为 STOP 状态。

Figure 92. START 与 STOP 状态

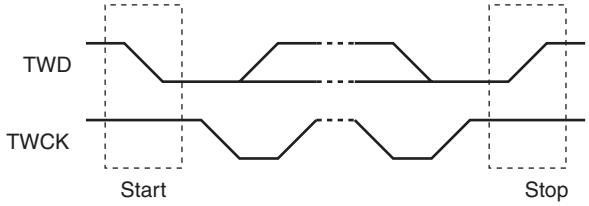
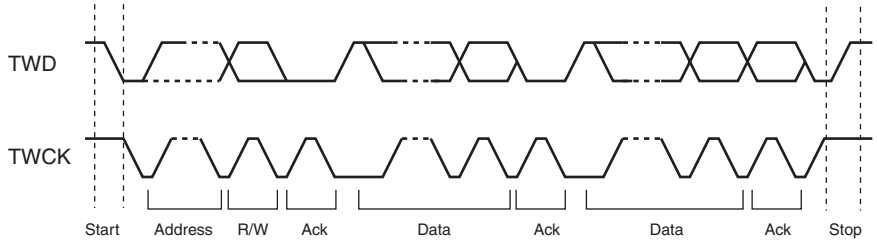


Figure 93. 传输格式



工作模式

TWI 有两种工作模式：

- 主机发送模式
- 主机接收模式

主机模式下， TWI 控制寄存器 (TWI_CR) 可配置为接口。该模式下，根据在时钟波形发生器寄存器 (TWI_CWGR) 中编程值产生时钟。该寄存器定义定义了 TWCK 信号，使能接口以适应宽范围时钟。

数据发送

主机初始化 Start 状态后，向主机模式寄存器 (TWI_MMR 中 DADR) 发送一个 7 位从机地址，以通知从机器件。从机地址后的位表示传输方向 (写或读)。该位为 0，说明是写操作 (发送操作) ；若该位为 1，说明为数据读请求 (接收操作)。

TWI 传输要求从机每收到一个字节后均要给出应答。在应答时钟脉冲中，主机释放数据线 (HIGH)，将从机拉低以产生应答。主机在该时钟脉冲中轮询数据线，若从机未应答该字节将置位状态寄存器的 **NAK** 位。与其它状态位相同，若使能中断使能寄存器 (TWI_IER) 将产生中断。写发送保持寄存器 (TWI_THR) 后，设置控制寄存器的 START 位以启动传输。数据在内部移位寄存器中移位，当检测到应答，TXRDY 位置位，直到 TWI_THR 中有新数据写入，才清除该位 (见 Figure 95)。主机产生 STOP 状态来结束传输。

设置 START 后开始读序列。当状态寄存器中 RXRDY 位置位时，接收保持寄存器 (TWI_RHR) 以收到一个字符。当读 TWI_RHR 时 RXRDY 位复位。

TWI接口可执行多种传输格式 (7位从机地址，10位从机地址)。通过主机模式寄存器(TWI_MMR) 配置三个内部地址字节。若从机仅支持 7 位地址，**IADRSZ** 必须置为 0。若从机地址大于 7 位，用户必须配置地址大小 **(IADRSZ**) 并在内部地址寄存器 (TWI_IADR) 中设置其它从机地址位。

Figure 94. 1、 2 或 3 字节内部地址及 1 字节数据的主机写操作

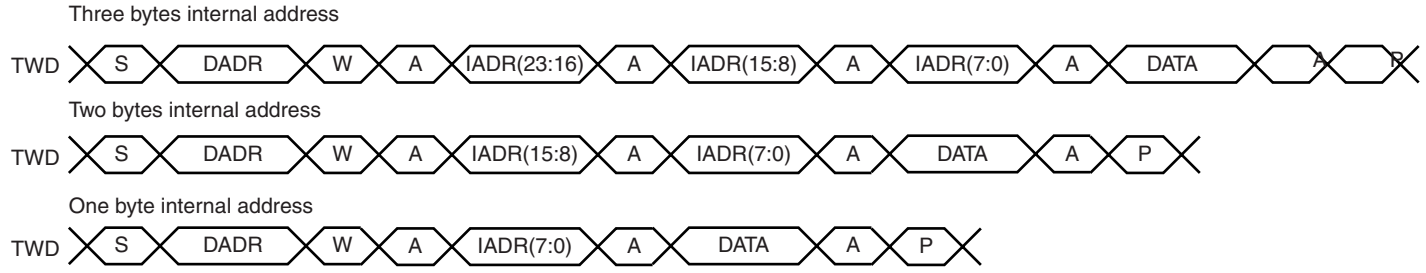


Figure 95. 1 字节内部地址及多数据字节的主机写操作

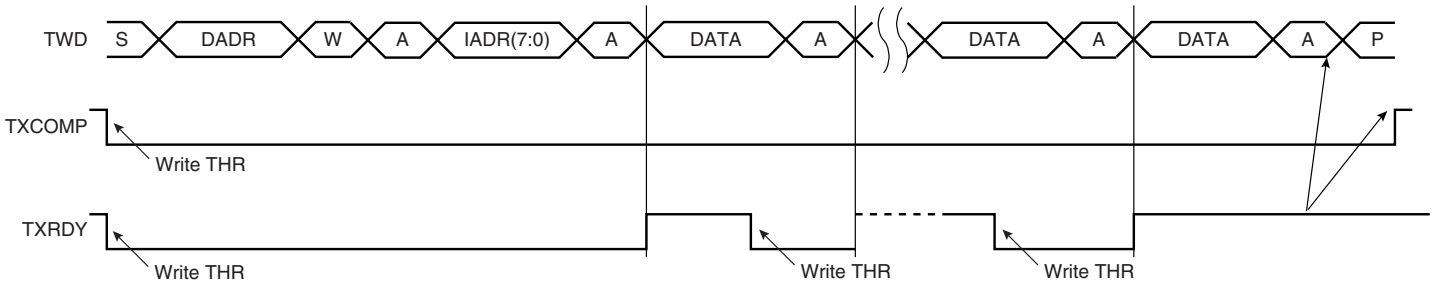


Figure 96. 1、 2 或 3 字节内部地址及 1 字节数据的主机读操作

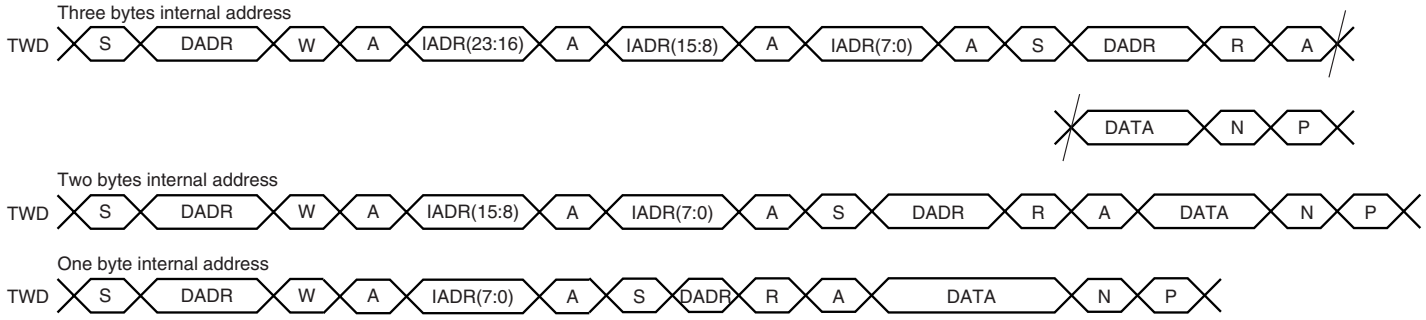
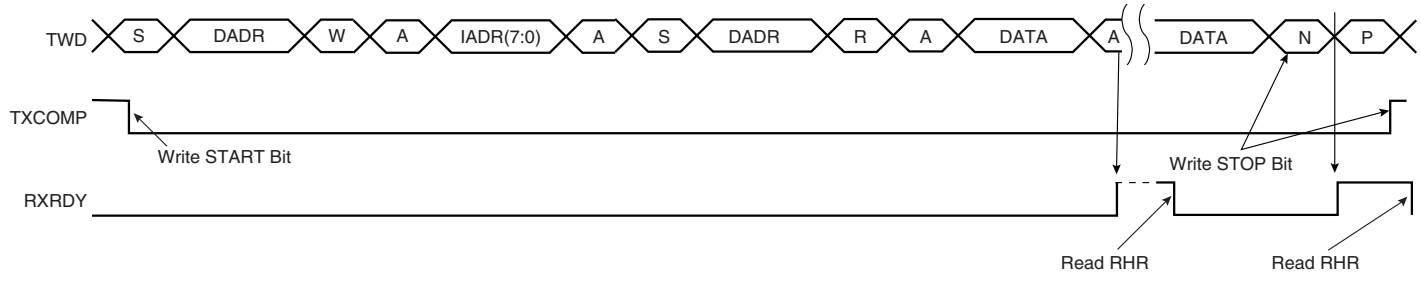


Figure 97. 1 字节内部地址及多数据字节的主机读操作

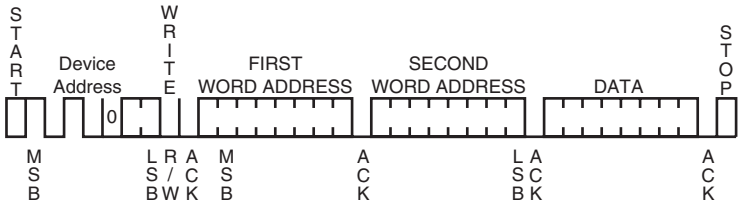


- S = Start
- P = Stop
- W = 读 / 写

- A = 应答
- DADR= 器件地址
- IADR = 内部地址

Figure 98 给出了在 Atmel AT24LC512 EEPROM 中写入一字节的图示。该图示范了如何使用内部地址访问器件。

Figure 98. 内部地址用法



读 / 写流程图

Figure 99 on page 260 与 Figure 100 on page 261 中流程图给出主机模式下读、写操作示例。可使用轮询或中断方式来检验状态位。使用中断方式时要先配置中断使能寄存器 (TWI_IER)。

Figure 99. 主机模式下 TWI 写操作

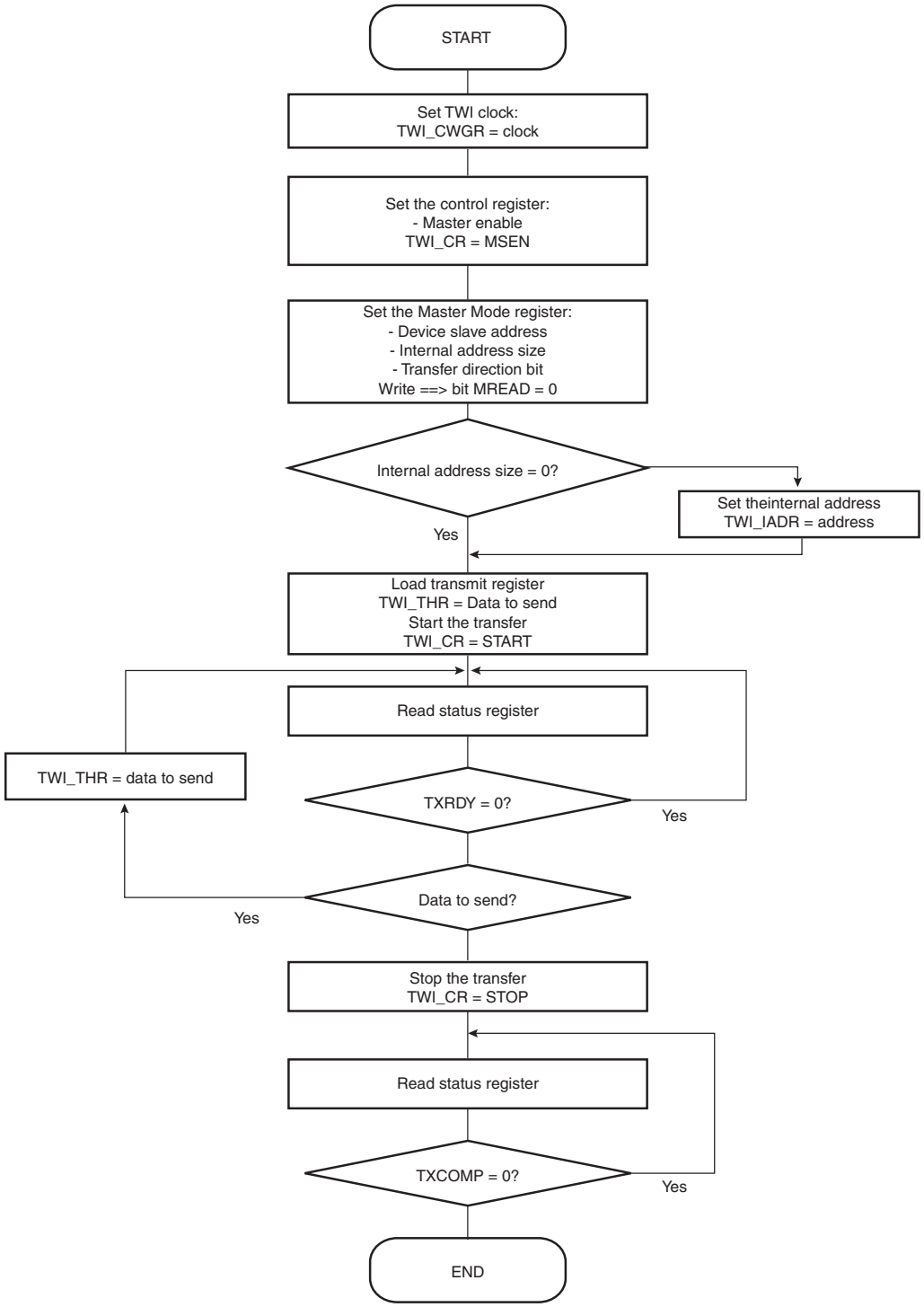
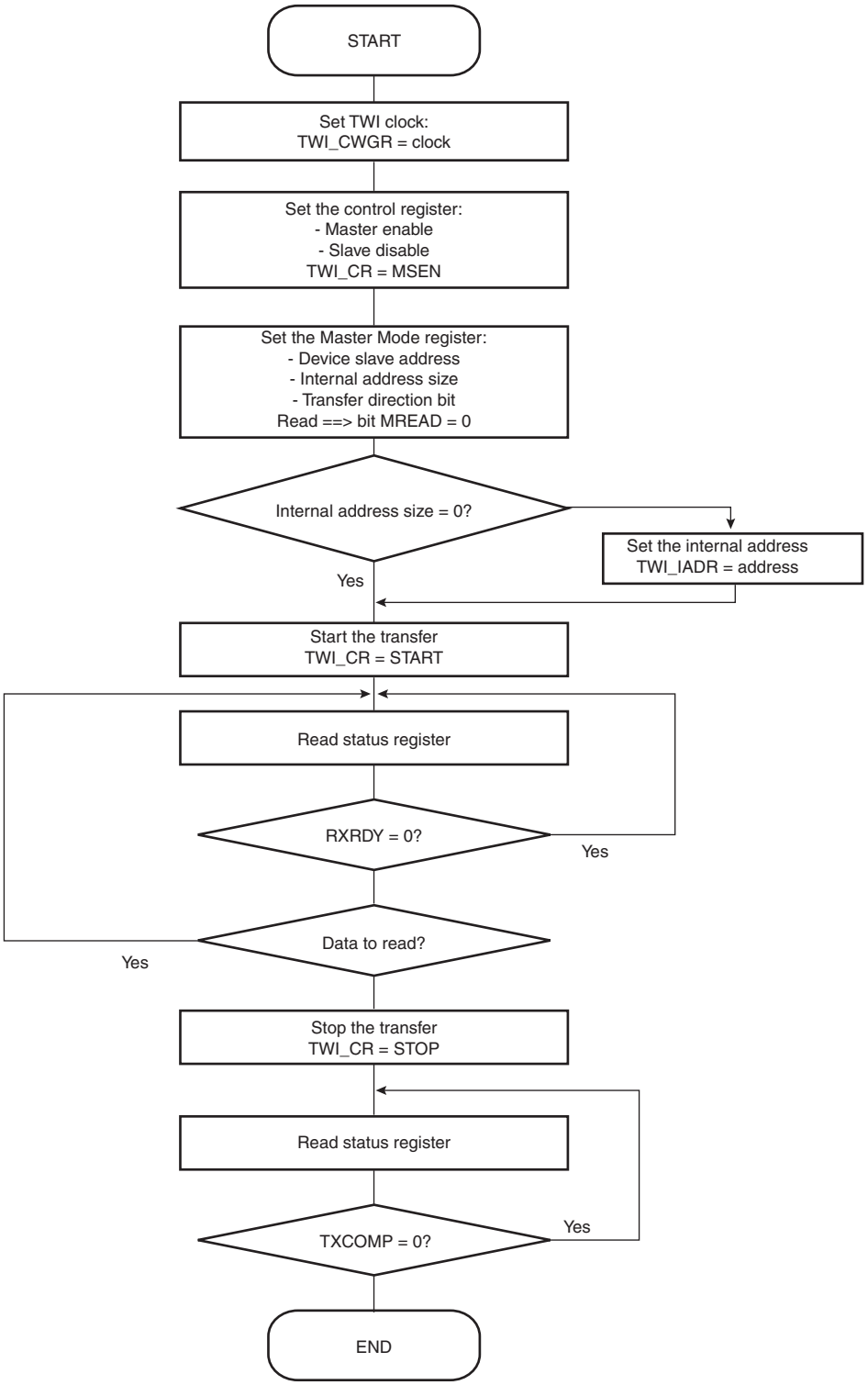


Figure 100. 主机模式下 TWI 读操作



两线接口 (TWI) 用户接口

Table 57. 两线接口 (TWI) 寄存器映射

偏移	寄存器	名称	访问类型	复位值
0x0000	控制寄存器	TWI_CR	只写	N/A
0x0004	主机模式寄存器	TWI_MMR	读 / 写	0x0000
0x0008	保留	–	–	–
0x000C	内部地址寄存器	TWI_IADR	读 / 写	0x0000
0x0010	时钟波形发生寄存器	TWI_CWGR	读 / 写	0x0000
0x0020	状态寄存器	TWI_SR	只读	0x0008
0x0024	中断使能寄存器	TWI_IER	只写	N/A
0x0028	中断禁用寄存器	TWI_IDR	只写	N/A
0x002C	中断屏蔽寄存器	TWI_IMR	只读	0x0000
0x0030	接收保持寄存器	TWI_RHR	只读	0x0000
0x0034	发送保持寄存器	TWI_THR	读 / 写	0x0000
0x0038-0x00FC	保留	–	–	–

TWI 控制寄存器

寄存器名称： TWI_CR

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
SWRST	–	–	–	MSDIS	MSEN	STOP	START

- **START: 发送 START 状态**

0 = 无效。

1 = 根据模式寄存器定义的特性，帧传输是以发送 START 位开始的。

当 TWI 外设需要由从机读数据时该步骤必须执行。当在主机模式下配置一个写操作，用户在保持寄存器写入一个字符，则使用模式寄存器开始发送一帧数据。

- **STOP: 发送 STOP 状态**

0 = 无效。

1 = 主机读或写模式下，当前字节传输完成后，发送 STOP 状态。

单数据字节主机读写时， START 与 STOP 必须设置。

多数据字节主机读写时， STOP 在传输 ACK/NACK 位前必须设置。

主机读模式下，若收到 NACK 位， STOP 自动执行。

多数据字节写操作时，当 THR 与移位寄存器均为空时，自动发送 STOP 状态。

- **MSEN: TWI 主机传输使能**

0 = 无效。

1 = 若 MSDIS = 0，主机数据传输使能。

- **MSDIS: TWI 主机传输禁用**

0 = 无效。

1 = 主机数据传输禁用，发送所有挂起数据。写操作时，发送移位和保持字符 (若其中包含数据) ；读操作时，在禁用前接收所有传输字符。

- **SWRST: 软件复位**

0 = 无效。

1 = 与系统复位等效。

TWI 主机模式寄存器

寄存器名称： TWI_MMR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	DADR						
15	14	13	12	11	10	9	8
–	–	–	MREAD	–	–	IADRSZ	
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **IADRSZ: 内部器件地址长**

IADRSZ[9:8]		
0	0	无内部器件地址
0	1	1 字节内部器件地址
1	0	2 字节内部器件地址
1	1	3 字节内部器件地址

- **MREAD: 主机读方向**

0 = 主机写。

1 = 主机读。

- **DADR: 器件地址**

器件地址用于在主机模式读或写下访问从机器件。

TWI 内部地址寄存器

寄存器名称： TWI_IADR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
IADR							
15	14	13	12	11	10	9	8
IADR							
7	6	5	4	3	2	1	0
IADR							

- **IADR: 内部地址**

0、 1、 2 或 3 字节由 IADRSZ 决定。

TWI 时钟波形发生器寄存器

寄存器名称： TWI_CWGR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	CKDIV		
15	14	13	12	11	10	9	8
CHDIV							
7	6	5	4	3	2	1	0
CLDIV							

- **CLDIV: 时钟低分频器**

SCL 低周期定义如下：

$$T_{low} = ((\text{CLDIV} \times 2^{\text{CKDIV}}) + 3) \times T_{MCK}$$

- **CHDIV: 时钟高分频器**

SCL 高周期定义如下：

$$T_{high} = ((\text{CHDIV} \times 2^{\text{CKDIV}}) + 3) \times T_{MCK}$$

- **CKDIV: 时钟分频器**

SCL 高低周期中 CKDIV 均增加。

TWI 状态寄存器

寄存器名称： TWI_SR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	NACK
7	6	5	4	3	2	1	0
UNRE	OVRE	–	–	–	TXRDY	RXRDY	TXCOMP

- **TXCOMP: 传输完成**

0 = 对于主机，为当前帧长；对于从机，为从收到 START 到收到 STOP。

1 = 当保持与移位寄存器均为空且 STOP 状态已发送 (主机) 或已收到 (从机)，或 MSEN 置位 (使能 TWI)。

- **RXRDY: 接收保持寄存器就绪**

0 = 上次 TWI_RHR 读操作后未收到字符。

1 = 上次 TWI_RHR 读操作后收到一字符。

- **TXRDY: 发送保持寄存器就绪**

0 = 发送保持寄存器中内容未传输到移位寄存器中，当对 TWI_THR 寄存器写入时设置为 0。

1 = 当数据由TWI_THR传入内部移位寄存器或检测到NACK错误，TXRDY与TXCOMP及NACK同时置位。当MSEN置位时，TXRDY 同样置位 (使能 TWI)。

- **OVRE: 溢出错误**

0 = RXRDY 已置位，而 TWI_RHR 未载入数据。

1 = RXRDY 置位时， TWI_RHR 已载入数据。当 TXCOMP 置位时，通过读 TWI_SR 进行复位。

- **UNRE: 空栈读出错**

0 = 无空栈读出错。

1 = 当移位寄存器加载数据时，TWI_THR 中无有效数据 (TXRDY 置位)。主机模式下，将自动产生 STOP 位。当 TXCOMP 置位时，通过读 TWI_SR 进行复位。

- **NACK: 无应答**

0 = 每个数据字节均被远端 TWI 从机正确接收。

1 = 从机未给出应答。与 TXCOMP 同时置位。读后复位。

TWI 中断使能寄存器

寄存器名称： TWI_IER

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	NACK
7	6	5	4	3	2	1	0
UNRE	OVRE	–	–	–	TXRDY	RXRDY	TXCOMP

- **TXCOMP: 传输完成**
- **RXRDY: 接收保持寄存器就绪**
- **TXRDY: 发送保持寄存器就绪**
- **OVRE: 溢出错误**
- **UNRE: 空栈读错误**
- **NACK: 无应答**

0 = 无效。

1 = 使能相应中断。

TWI 中断禁用寄存器

寄存器名称 :TWI_IDR

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	NACK
7	6	5	4	3	2	1	0
UNRE	OVRE	–	–	–	TXRDY	RXRDY	TXCOMP

- **TXCOMP: 传输完成**
- **RXRDY: 接收保持寄存器就绪**
- **TXRDY: 发送保持寄存器就绪**
- **OVRE: 溢出错误**
- **UNRE: 空栈读错误**
- **NACK: 无应答**

0 = 无效。

1 = 禁用相应中断。

TWI 中断屏蔽寄存器

寄存器名称： TWI_IMR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	NACK
7	6	5	4	3	2	1	0
UNRE	OVRE	–	–	–	TXRDY	RXRDY	TXCOMP

- **TXCOMP: 传输完成**
- **RXRDY: 接收保持寄存器就绪**
- **TXRDY: 发送保持寄存器就绪**
- **OVRE: 溢出错误**
- **UNRE: 空栈读错误**
- **NACK: 无应答**

0 = 相应中断禁用。

1 = 相应中断使能。

TWI 接收保持寄存器

寄存器名称 :TWI_RHR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
RXDATA							

- **RXDATA: 主机或从机接收保持数据**

TWI 发送保持寄存器

寄存器名称： TWI_THR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
TXDATA							

- **TXDATA: 主机或从机发送保持数据**

通用同步 / 异步收发器 (USART)

概述

通用同步异步收发器 (USART) 提供一个全双工通用同步异步串行连接。数据帧格式可编程 (数据长度，奇偶校验位，停止位数) 以支持尽可能多的标准。接收器执行奇偶错误、帧错误及溢出错误检测。接收器超时使能可变长帧处理，发送器时间保障方便与慢慢速远程器件通信。接收与发送地址位也支持多点通信。

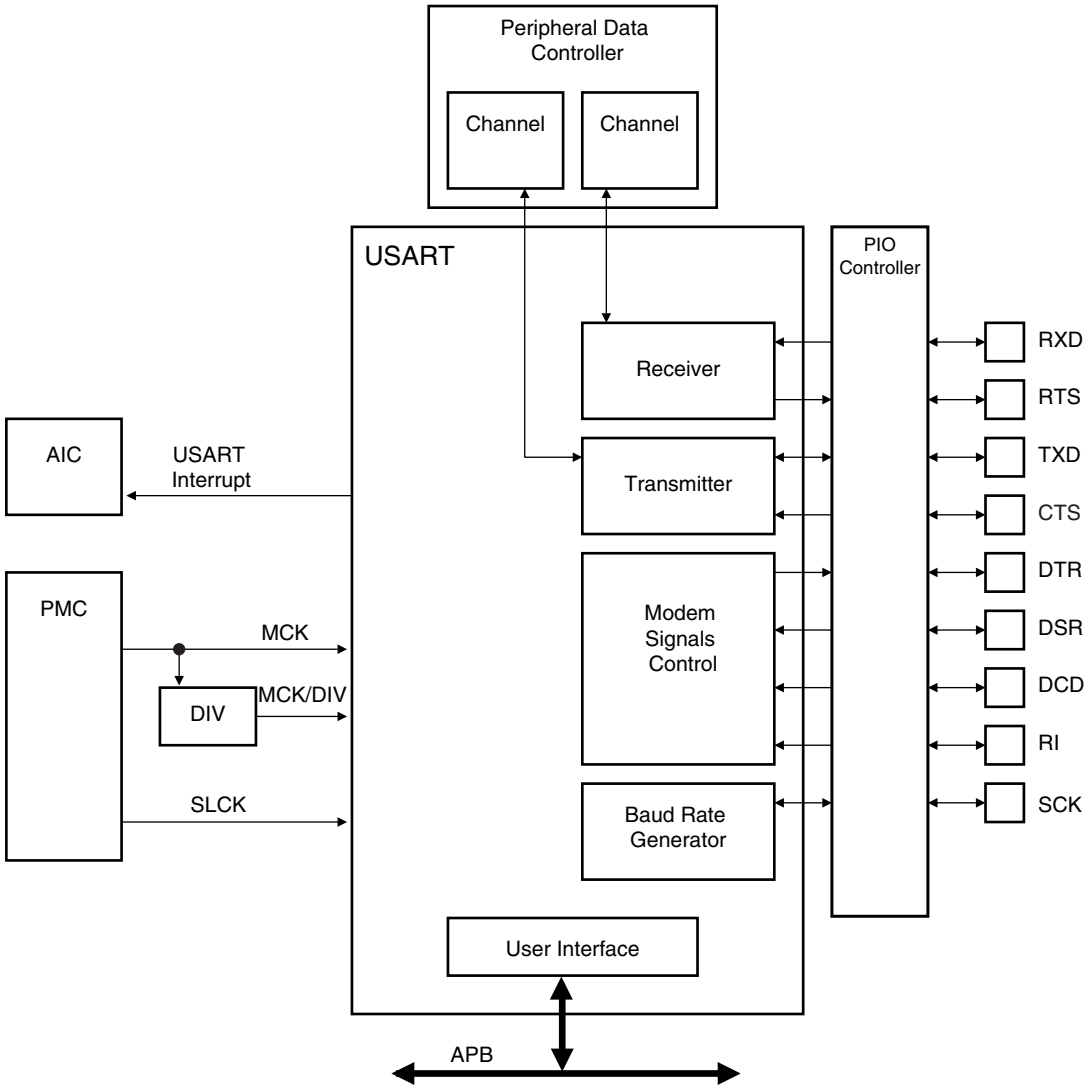
USART 有三种测试模式：远程回环、本地回环及自动回应。

USART 支持 RS485 总线提供的特殊操作模式，通过 ISO7816 T = 0 或 T = 1 智能卡插槽、红外收发器并与调制解调器连接。硬件握手通信通过 RTS 与 CTS 引脚自动管理溢出控制。

USART 支持与使能由发送器到接收器的数据传输的外设数据控制器的连接。PDC 提供没有处理器干扰的链缓冲管理。

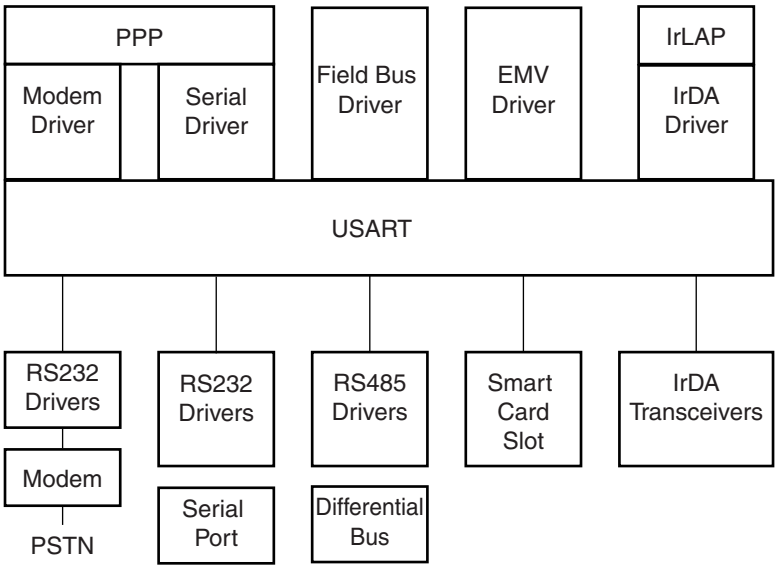
方框图

Figure 101. USART 框图



应用框图

Figure 102. 应用框图



I/O 线说明

Table 58. I/O 线说明

名称	说明	类型	有效电平
SCK	串行时钟	I/O	
TXD	发送串行数据	I/O	
RXD	接收串行数据	输入	
RI	环指示器	输入	低
DSR	数据设置就绪	输入	低
DCD	数据载波检测	输入	低
DTR	数据中止就绪	输出	低
CTS	发送清除	输入	低
RTS	发送请求	输出	低

附属产品

I/O 线

连接 USART 的引脚可与 PIO 线复用。必须先对 PIO 编程以将 USART 引脚分配到期望的外设功能。若 USART 的 I/O 线未使用， PIO 控制器可将其用作其它功能。

调制解调器的引脚不一定在 USART 中执行。通常，只有 USART1 是完全为调制解调器信号准备的。对于其它 USART 则没有相应引脚，相关控制位及状态对 USART 无影响。

电源管理

USART 时钟不连续。在使用 USART 前，必须先使能电源管理控制器 (PMC) 中的 USART 时钟。但若应用中不需要 USART 时，USART 时钟可停止，并在需要时重新运行。该情况下，USART 恢复到停止前的状态。

对 USART 配置不需要使能 USART 时钟。

中断

USART中断线与高级中断控制器的一个内部中断源连接。使用USART中断请求前要先对AIC编程。注意，边沿敏感模式下不推荐使用 USART 中断线。

功能说明

USART 可管理多类型串行同步或异步通信。

它支持下列通信模式：

- 5 到 9 位全双工异步串行通信：
  - 高位或低位在先
  - 1、 1.5 或 2 位停止位
  - 奇检验、偶检验、标志、间隔或无
  - 接收器频率 8 或 16 倍重采样
  - 可选硬件握手
  - 可选调试解调器信号管理
  - 可选间断管理
  - 可选多点串行通信
- 高速 5 到 9 位全双工串行通信
  - 高位或低位在先
  - 1 或 2 位停止位
  - 奇检验、偶检验、标志、间隔或无
  - 接收器频率 8 或 16 倍重采样
  - 可选硬件握手
  - 可选调试解调器信号管理
  - 可选间断管理
  - 可选多点串行通信
- 含驱动器控制信号的 RS485
- ISO7816，与智能卡连接的 T0 或 T1 协议
  - NACK 处理，有复制与反复限制的错误计数器
- 红外 IrDA 调制解调
- 测试模式
  - 远程回环、本地回环、自动回应

波特率发生器

波特率发生器给接收器与发送器提供名为波特率时钟的位周期时钟。

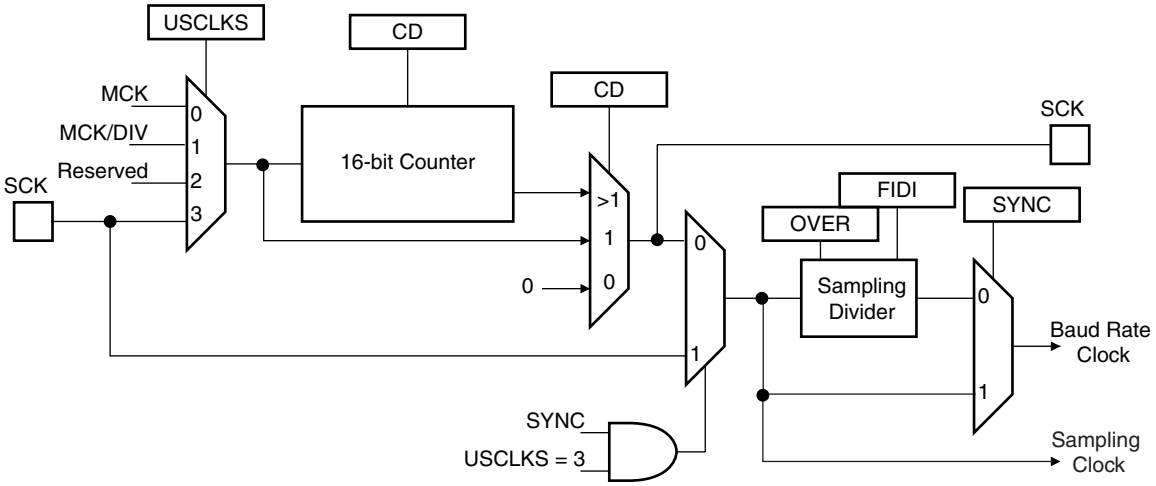
波特率发生器时钟源可设置模式寄存器 (US_MR) 的 USCLKS 域为下列状态：

- 主机时钟 MCK
- 主机时钟分频，分频因子由产品确定，通常情况下为 8
- 外部时钟，在 SCK 引脚有效

波特率发生器是基于波特率发生器寄存器 (US_BRGR) 中 CD 域编程的 16 位分频器。若 CD 为 0，波特率发生器不产生时钟；若 CD 为 1，分频器被绕过，并失效。

若选择外部 SCK 时钟，SCK 引脚高低电平持续时间必须比一个主机时钟 (MCK) 周期长。MCK 频率至少是 SCK 上信号频率的 4.5 倍。

Figure 103. 波特率发生器



异步模式下的波特率

若 USART 工作在异步模式下，选定的时钟首先除以 US_BRGR 寄存器中 CD 域的值。所得到的时钟作为接收器的采样时钟，再根据 US_MR 寄存器中的 OVER 位决定被 16 或 8 分频。

若 OVER 为 1，接收器采样时钟为波特率时钟的 8 倍；若 OVER 清零，接收器采样时钟为波特率时钟的 16 倍。

下列公式用来计算波特率：

$$Baudrate = \frac{SelectedClock}{(8(2-Over)CD)}$$

假设 MCK 工作再最高时钟频率下，且 OVER 为 1，上式给出了 MCK 8 分频后的最大波特率。

波特率计算示例

Table 59给出在不同时钟源频率下得到波特率为38400的CD计算方法。表中同时也给出了实际结果及误差。

Table 59. 波特率示例 (OVER = 0)

源时钟	期望波特率	计算结果	CD	实际波特率	误差
MHz	Bit/s			Bit/s	
3 686 400	38 400	6.00	6	38 400.00	0.00%
4 915 200	38 400	8.00	8	38 400.00	0.00%
5 000 000	38 400	8.14	8	39 062.50	1.70%
7 372 800	38 400	12.00	12	38 400.00	0.00%
8 000 000	38 400	13.02	13	38 461.54	0.16%
12 000 000	38 400	19.53	20	37 500.00	2.40%
12 288 000	38 400	20.00	20	38 400.00	0.00%
14 318 180	38 400	23.30	23	38 908.10	1.31%
14 745 600	38 400	24.00	24	38 400.00	0.00%
18 432 000	38 400	30.00	30	38 400.00	0.00%
24 000 000	38 400	39.06	39	38 461.54	0.16%
24 576 000	38 400	40.00	40	38 400.00	0.00%

Table 59. 波特率示例 (OVER = 0)

源时钟	期望波特率	计算结果	CD	实际波特率	误差
25 000 000	38 400	40.69	40	38 109.76	0.76%
32 000 000	38 400	52.08	52	38 461.54	0.16%
32 768 000	38 400	53.33	53	38 641.51	0.63%
33 000 000	38 400	53.71	54	38 194.44	0.54%
40 000 000	38 400	65.10	65	38 461.54	0.16%
50 000 000	38 400	81.38	81	38 580.25	0.47%
60 000 000	38 400	97.66	98	38 265.31	0.35%
70 000 000	38 400	113.93	114	38 377.19	0.06%

波特率由下式计算得到：

$$BaudRate = MCK / CD \times 16$$

波特率误差由下式计算得到。建议误差应低于 5%。

$$Error = 1 - \left(\frac{ExpectedBaudRate}{ActualBaudRate}\right)$$

同步模式下的波特率

若 USART 工作在同步模式下，选定的时钟除以 US_BRGR 寄存器中 CD 域的值。

$$BaudRate = \frac{SelectedClock}{CD}$$

同步模式下，若选择外部时钟 (USCLKS = 3)，时钟由 USART SCK 引脚信号提供。不需分频。写入 US_BRGR 中的值无效。系统时钟频率至少是外部时钟频率的 4.5 倍。

当同时选择外部时钟 SCK 与内部分频时钟 (MCK/DIV) 时，若用户要 SCK 引脚信号占空比为 50:50，则 CD 值为偶数。若选择内部时钟 MCK，即使 CD 值为奇数， SCK 引脚信号占空比为 50:50。

ISO 7816 模式下波特率

ISO7816 用下式定义比特率：

$$B = \frac{Di}{Fi} \times f$$

其中：

- B 为比特率
- Di 为比特率调整因子
- Fi 为时钟频率分频因子
- f 为 ISO7816 时钟频率 (Hz)

Di 是一个 4 位二进制值，称为 DI，见 Table 60。

Table 60. D 的二进制与十进制值

DI 域	0001	0010	0011	0100	0101	0110	1000	1001
Di (十进制)	1	2	4	8	16	32	12	20

Fi 是一个 4 位二进制值，称为 FI，见 Table 61。

Table 61. Fi 的二进制与十进制值

FI 域	0000	0001	0010	0011	0100	0101	0110	1001	1010	1011	1100	1101
Fi (十进制	372	372	558	744	1116	1488	1860	512	768	1024	1536	2048

Table 62 给出 Fi/Di 比值，在 ISO7816 时钟与波特率时钟间。

Table 62. 可能的 Fi/Di 比值

Fi/Di	372	558	774	1116	1488	1806	512	768	1024	1536	2048
1	372	558	744	1116	1488	1860	512	768	1024	1536	2048
2	186	279	372	558	744	930	256	384	512	768	1024
4	93	139.5	186	279	372	465	128	192	256	384	512
8	46.5	69.75	93	139.5	186	232.5	64	96	128	192	256
16	23.25	34.87	46.5	69.75	93	116.2	32	48	64	96	128
32	11.62	17.43	23.25	34.87	46.5	58.13	16	24	32	48	64
12	31	46.5	62	93	124	155	42.66	64	85.33	128	170.6
20	18.6	27.9	37.2	55.8	74.4	93	25.6	38.4	51.2	76.8	102.4

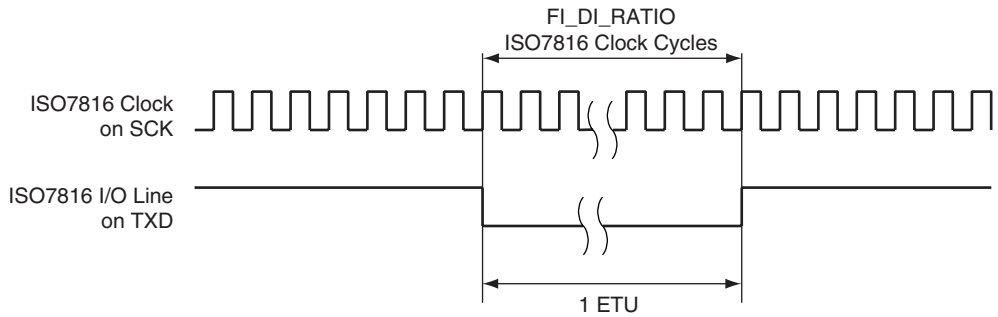
若 USART 配置为 ISO7816 模式，模式寄存器 (US_MR) 中的 USCLKS 域指定的时钟首先除以波特率发生器寄存器 (US_BRGR) 中的 CD 编程值。得到的时钟提供给 SCK 引脚，作为智能卡时钟输入。即 CLKO 位可在 US_MR 中设置。

该时钟由 FI_DI_Ratio 寄存器 (US_FIDI) 中的 FI_DI_RATIO 域值分频。由 ISO7816 模式下可除高达 2047 的采样除法器执行。Fi/Di 比率必须为整数，且用户必须尽量将 FI_DI_RATIO 值接近于期望值。

FI_DI_RATIO 域复位值为 0x174 (十进制 372) ,这是 ISO7816 时钟与比特率间公共的除数 (Fi = 372， Di = 1)。

Figure 104 给出了与位时钟与 ISO 7816 时钟相关的关系。

Figure 104. 基本时间单元 (ETU)



接收器与发送器控制

复位后接收器禁用。用户必须通过设置控制寄存器 (US_CR) 的 RXEN 位使能接收器。但接收器寄存器在接收器时钟使能前可编程。

复位后发送器禁用。用户必须通过设置控制寄存器 (US_CR) 的 TXEN 位使能发送器。但发送器寄存器在发送器时钟使能前可编程。

发送器与接收器可一起或分别使能。

任意时刻，软件可通过分别置位 US_CR 寄存器中的 RSTRX 与 RSTTX 执行 USART 接收器或发送器复位。软件复位与硬件复位效果相同。复位时，不管是接收器还是发送器，通信立即停止。

用户也可通过设置 US_CR 寄存器中的 RXDIS 与 TXDIS 独立禁用接收器与发送器。若在接收字符时接收器禁用，USART 等待当前字符接收结束，再停止接收。若发送器工作时禁用，USART 等待当前字符及存于发送编程寄存器 (US_THR) 中的字符发送完成 。若编程设置了时间保障，它将正常处理。

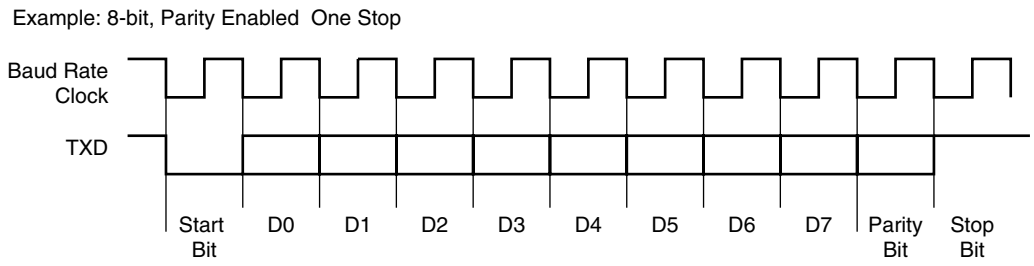
同步与异步模式

发送器操作

在同步与异步模式下 (SYNC = 0 或 SYNC = 1)，发送器操作相同。 1 个起始位，最多 9 个数据位， 1 个可选的奇偶校验位及最多 2 个停止位，在每个串行设置下降沿由 TXD 引脚移出。

数据位数目由 US_MR 寄存器的 CHRL 域及 MODE9 位决定。设置 MODE 9 位时，不管 CHRL 域设置，数据位为 9 位。奇偶校验位由 PAR 域设置。可配置为奇检验、偶检验、空检验、标志检验或无校验位。MSBF 域配置首先发送的位。若写入 1，将先发送最高位；若写入 0，将先发送最低位。停止位数目由 NBSTOP 域选择。异步模式下支持 1.5 停止位。

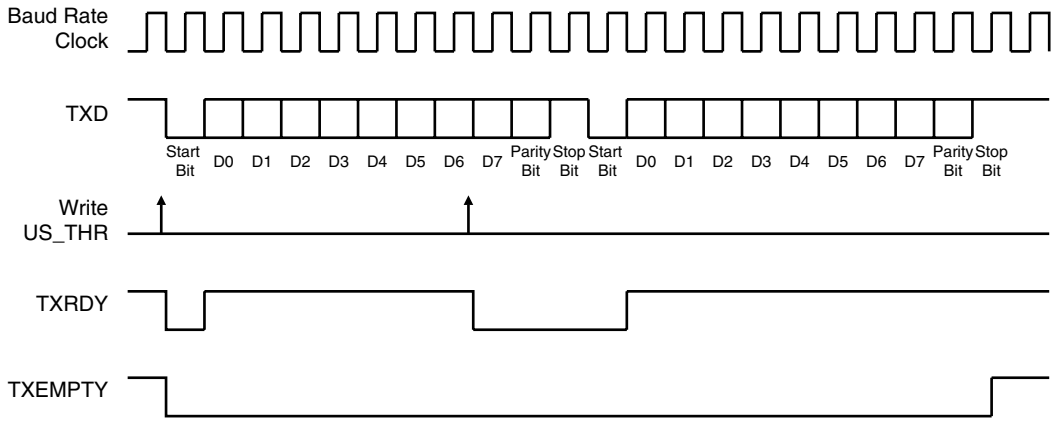
Figure 105. 字符发送



字符发送到发送保持寄存器 (US_THR) 中。发送器在通道状态寄存器 (US_CSR) 中有 2 个状态位：TXRDY (发送就绪) 表示 US_THR 空且 TXEMPTY，即所有写入 US_THR 中的字符已开始处理。当当前字符处理完成，最后写入US_THR的字符送入发送器移位寄存器中，且US_THR变空，因此 TXRDY 升高。

发送器禁用后， TXRDY 与 TXEMPTY 位均为低。当 TXRDY 有效，在 US_THR 中写入字符无效，且写入数据丢失。

Figure 106. 发送器状态



异步接收器

若 USART 工作在异步模式下 (SYNC = 0)，接收器对 RXD 输入线重采样。重采样为 16 或 8 倍波特率时钟，由 US_MR 中的 OVER 位设置。

接收器对 RXD 线采样。若在 1.5 个比特时间内采样值为 0，即表示检测到起始位，然后数据、校验位、停止位等以比特速率时钟采样。

若重采样为 16(OVER 为 0)， 8 次采样结果均为 0，表示检测到起始位。然后，每 16 个采样时钟周期中对数据、校验位、停止位依次采样。若重采样为 8 (OVER 为 1)， 4 次采样结果均为 0，表示检测到起始位。然后，每 8 个采样时钟周期中对数据、校验位、停止位依次采样。

数据位数、最先发送位及校验模式选择的域及位与发送器相同，即分别为 CHRL、 MODE9、MSBF 及 PAR。停止位数对接收器无效，因为无论 NBSTOP 域为何值，接收器都只确认 1 个停止位，因此会出现发送器与接收器间的重同步。此外，接收器在检测到停止位后即开始寻找新的起始位，因此当发送器只有 1 个停止位时也能实现重同步。

Figure 107 与 Figure 108 给出当 USART 工作在异步模式下的起始检测及字符接收。

Figure 107. 异步起始检测

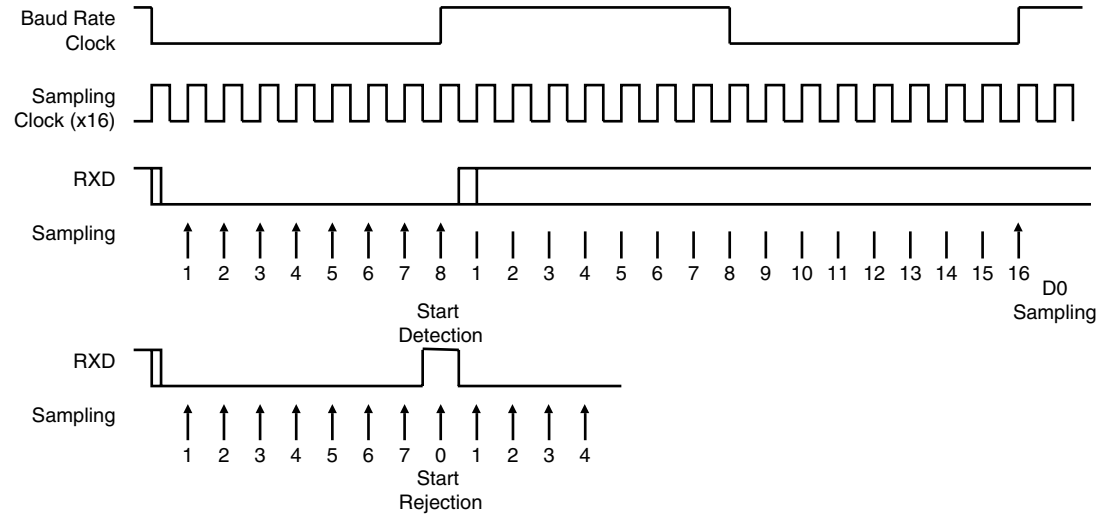
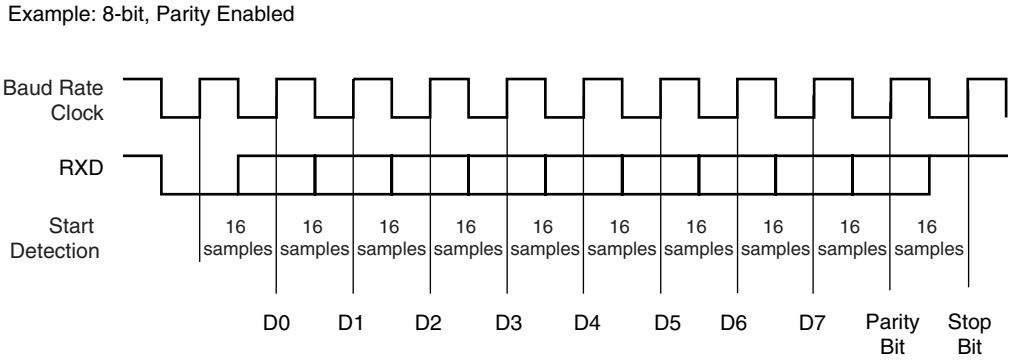


Figure 108. 异步字符接收



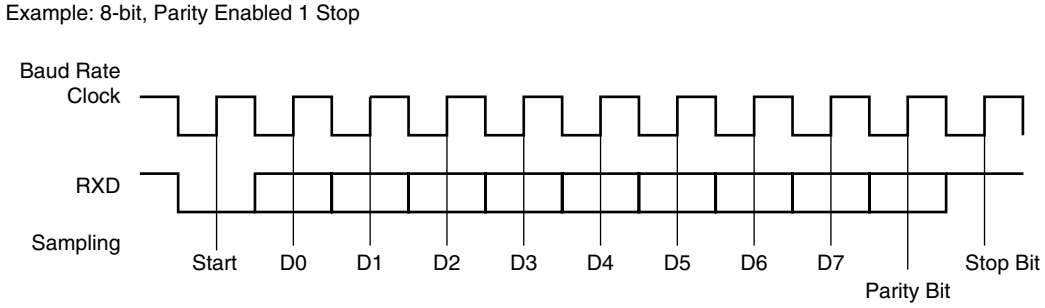
同步接收器

同步模式下 (SYNC = 1)，接收器在波特率时钟的每个上升沿对 RXD 信号采样。若检测到低电平，则为起始位。所有数据位、校验位及停止位依次采样，接收器等待下一个起始位。同步模式提供高速传输能力。

域及位的配置与异步模式下相同。

Figure 109 给出同步模式下的字符接收。

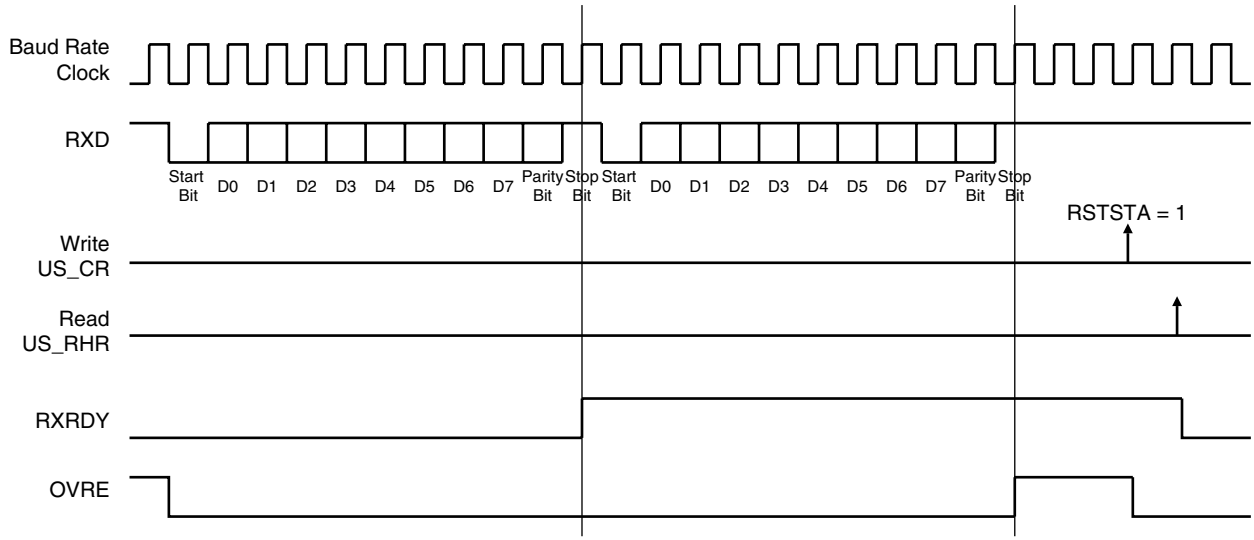
Figure 109. 同步模式字符接收



接收器操作

当字符接收完成，它将传输到接收保持寄存器 (US_RHR)，且状态寄存器 (US_CSR) 的 RXRDY 位变高。若字符完成而 RXRDY 置位，OVRE (溢出错误) 位置位。最后的字符传输到 US_RHR 并覆盖上一个字符。对控制寄存器 (US_CR) 的 RSTSTA (复位状态) 位写 1 将清除 OVRE 位。

Figure 110. 接收器状态



奇偶检验

USART通过对模式寄存器(US_MR)PAR域的设置，可支持5种奇偶检验模式。PAR域还使能多点模式，这将在其它章节予以讨论 。产生奇偶检验位并支持错误检测。

若选择偶检验，当发送器发送 1 的数目为偶数时，校验位发生器产生校验位为 1 ；当发送器发送 1 的数目为奇数时，校验位发生器产生校验位为 0。相应的，接收器校验位检测器对收到的 1 计数，若与采样到的校验位不符，则报告奇偶检验错误。若选择奇检验，当发送器发送 1 的数目为偶数时，校验位发生器产生校验位为 0 ；当发送器发送 1 的数目为奇数时，校验位发生器产生校验位为 1。相应的，接收器校验位检测器对收到的 1 计数，若与采样到的校验位不符，则报告奇偶检验错误。若使用标志奇偶检验，对于所有字符，奇偶检验发生器都将奇偶校验位置 1。若接收器采样得到的校验位为 0，接收器校验位检测器报告检验错误。若使用空奇偶检验，对于所有字符，奇偶检验发生器都将奇偶校验位置 0。若接收器采样得到的校验位为 1，接收器校验位检测器报告检验错误。若奇偶检验禁用，发送器不产生校验位，接收器也不报告奇偶检验错误。

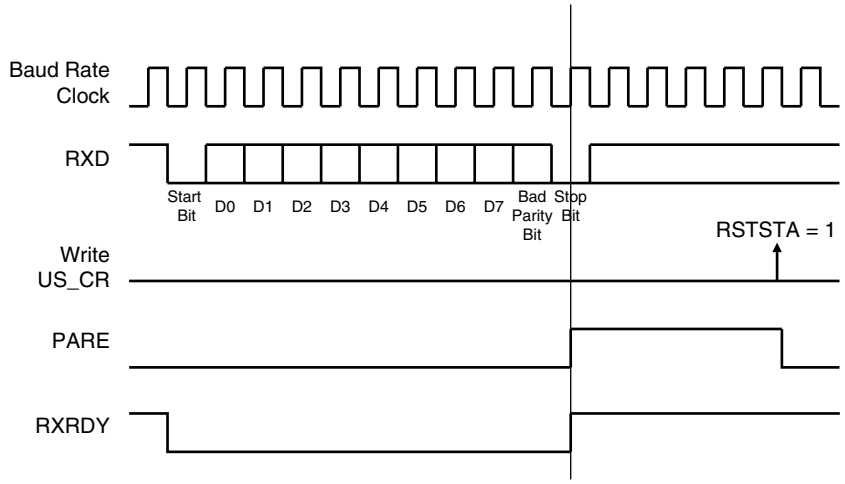
Table 63给出根据USART配置，字符0x41 (ASCII字符 “A”)的奇偶校验位的例子。由于有两位为 1，当奇偶检验为奇数时加 1 位，为偶数时加 0 位。

Table 63. 奇偶校验位示例

字符	十六进制	二进制	校验位	检验模式
A	0x41	0100 0001	1	奇检验
A	0x41	0100 0001	0	偶检验
A	0x41	0100 0001	1	标志检验
A	0x41	0100 0001	0	空检验
A	0x41	0100 0001	无	无

当接收器检测到奇偶检验误差，它设置通道状态寄存器 (US_CSR) 的 PARE (奇偶检验错误) 位。对控制寄存器 (US_CR) 的 RSTSTA 位写 1 将清除 PARE 位。 Figure 111 给出奇偶校验位的置位与清零。

Figure 111. 奇偶检验错误



多点模式

若模式寄存器 (US_MR) 的 PAR 域编程值为 0x6 或 0x7， USART 运行在多点模式下。该模式区分数据字符与地址字符。当奇偶校验位为 0 时发送数据；当奇偶校验位为 1 时发送地址。

若 USART 配置为多点模式，当奇偶校验位为高时，接收器将 PARE 位置位，当控制寄存器 SENDA 位为 1 时，若奇偶校验位为高，发送器可发送字符。

为处理奇偶检验错误，当控制寄存器 RSTSTA 位写 1 时， PARE 位清零。

当SENDA写入US_CR时，发送器发出地址字节(奇偶校验位置位)。此时，下一个写入US_THR的字节将作为地址来发送。没有写 SENDA 命令的任何写入 US_THR 的字符将正常被发送 (奇偶校验位为 0) 。

发送器时间保障

时间保障特性使能 USART 与慢速远程器件的连接。

时间保障功能使能发送器 TXD 线上在两字符间插入空闲状态。该空闲状态实际上是一个长停止位。

空闲状态的持续时间由发送时间保障寄存器 (US_TTGR) 的 TG 域编程设定。当该域编程值为零，则不产生时间保障。否则，在每次发送了停止位及 TG 中指定的位数周期后，在 TXD 线上发送器保持高电平。

如 Figure 112 所示， TXRDY 与 TXEMPTY 状态位的行为可由时间保障改变。 TXRDY 只有在下一字符起始位发送后才变高。若字符已写入 US_THR，则时间保障传输期间 TXRDY 保持为 0。由于时间保障是当前传输的一部分，因此 TXEMPTY 为低要保持到时间保障传输结束。

Figure 112. 时间保障操作

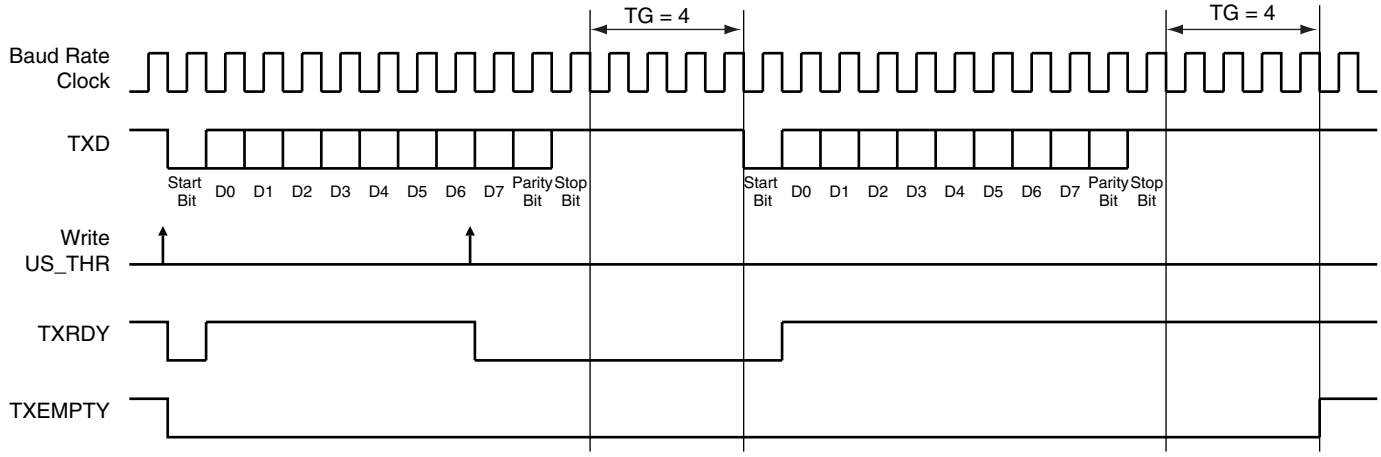


Table 64 列出对应不同波特率，发送器能处理的时间保障周期的最大值。

Table 64. 不同波特率下时间保障最大值

波特率	位时间	时间保障
Bit/sec	µs	ms
1 200	833	212.50
9 600	104	26.56
14400	69.4	17.71
19200	52.1	13.28
28800	34.7	8.85
33400	29.9	7.63
56000	17.9	4.55
57600	17.4	4.43
115200	8.7	2.21

接收器超时

接收器超时支持对长度可变的帧处理。该通信检测 RXD 线上的空闲状态。当检测到超时，通道状态寄存器 (US_CSR) 的 TIMEOUT 位变高并产生中断，以告知驱动器帧结束。

超时延迟周期 (接收器等待新字符时间) 在接收器超时寄存器 (US_RTOR) 的 TO 域编程。若 TO 域编程为 0，接收器超时禁用，将检测不到超时。 US_CSR 寄存器中 TIMEOUT 位保持为 0。否则，接收器将 TO 中值载入一个 16 位计数器。该计数器在每比特周期中自减并在收到新字符后重载。若计数器达到 0，状态寄存器中 TIMEOUT 位变高。

用户可：

- 至少收到一个字符后，当检测到超时时中断。通过在控制寄存器 (US_CR) 的 STTTO (启动超时) 位写 1 实现。
- 没有收到字符时周期中断。通过在 US_CR 中 RETTO (重载与启动超时) 位写 1 实现。

若 STTTO 执行，计数器时钟在收到第一个字符前停止。帧启动前 RXD 的空闲状态不提供超时。这样可防止周期性中断并在检测到 RXD 为空闲状态时使能帧结束等待。

若 RETTO 执行，计数器开始从 TO 值向下计数。因而产生周期性中断使可处理用户超时 ，例如当键盘上没有键入时。

Figure 113 给出接收器超时特性框图。

Figure 113. 计数器超时框图

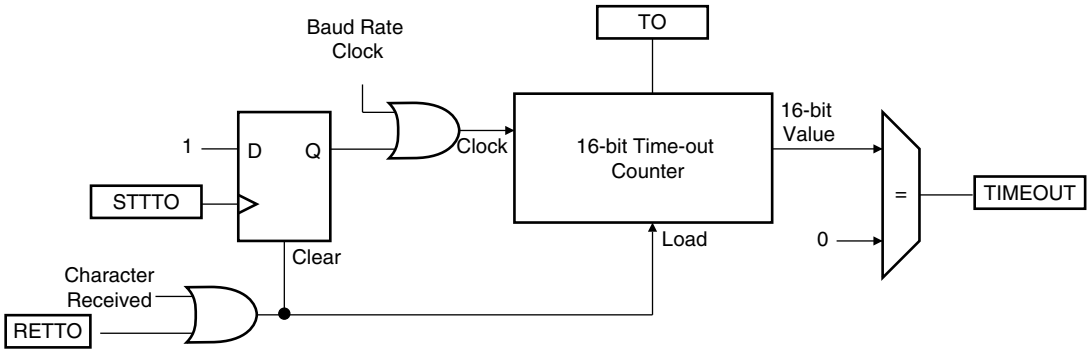


Table 65 给出某些标准波特率下的最大超时周期。

Table 65. 最大超时周期

波特率	位时间	超时
bit/sec	µs	ms
600	1 667	109 225
1 200	833	54 613
2 400	417	27 306
4 800	208	13 653
9 600	104	6 827
14400	69	4 551
19200	52	3 413
28800	35	2 276
33400	30	1 962

Table 65. 最大超时周期

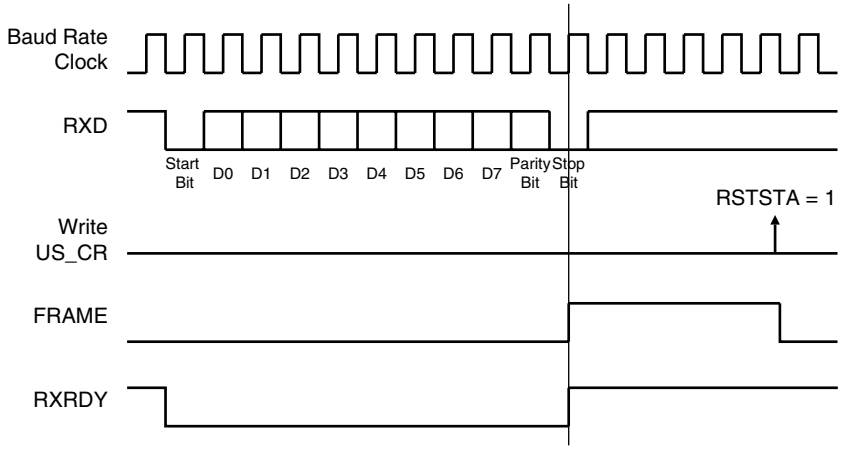
波特率	位时间	超时
56000	18	1 170
57600	17	1 138
200000	5	328

帧错误

接收器可检测帧错误。当检测到收到字符的停止位为 0 时产生帧错误。当接收器与发送器为完全同步时可能出现。

帧错误由 US_CSR 寄存器的 FRAME 位表示。检测到帧错误时，FRAME 位在停止位中间出现。通过将 US_CR 寄存器的 RSTSTA 位写为 1 将其清除。

Figure 114. 帧错误状态



发送间断

用户可请求发送器在 TXD 线上产生间断条件。使得至少在一个完整字符时间中 TXD 线为低。这与校验位及停止位均为 0 的 0x00 字符的发送情况相同。无论如何，发送器至少保证 TXD 线在一个完整的字符传输时间内为低，直到用户请求将间断条件删除。

将 US_CR 寄存器 STTBRK 位写 1 将发送一个间断。这可在任何时间执行，即使发送器为空 (移位寄存器及 US_THR 中均无字符) 或字符正在发送。若字符移出时出现间断请求，在 TXD 线变低前字符完成。

一旦需要 STTBRK 命令，在间断完成前忽略 STTBRK 命令。

通过在 US_CR 寄存器的 STPBRK 位写 1 将删除间断条件。若在最小间断持续时间 (一字符，包括起始位、数据位、校验位及停止位) 结束前请求 STPBRK，发送器保证间断条件完成。

发送器将间断视为一个字符，即只有当 US_CSR 寄存器中 TXRDY 为 1 时，STTBRK 与 STPBRK 命令才被考虑，如处理一个字符一样，间断条件启动将清除 TXRDY 与 TXEMPTY 位。

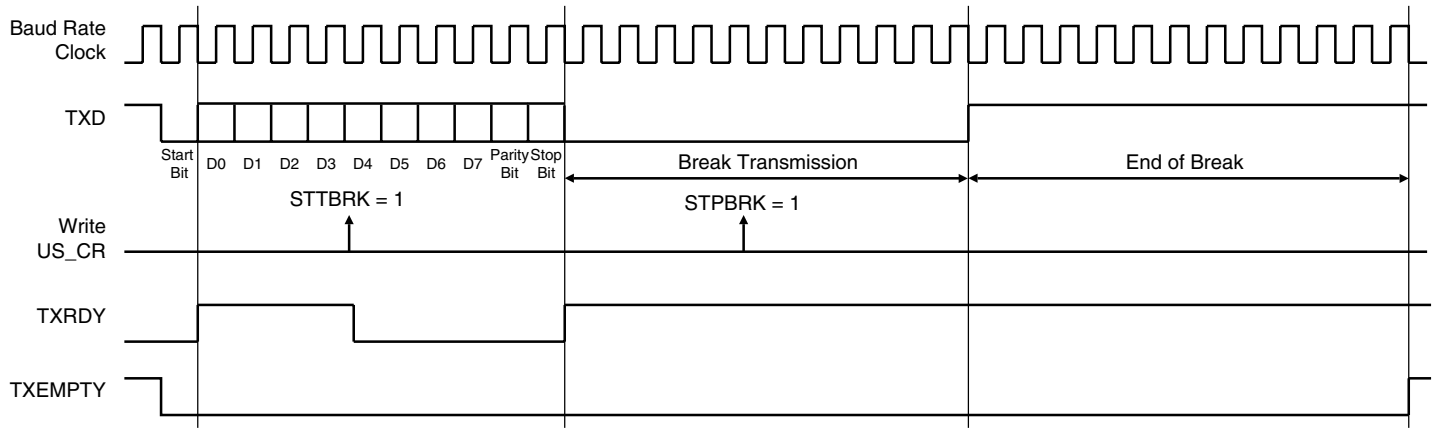
将 US_CR 寄存器中 STTBRK 与 STPBRK 位写为 1 结果无法预测。所有前面没有 STTBRK 命令的 STPBRK 命令请求将被忽略。当间断挂起时写入发送保持寄存器但未启动的字节将忽略。

间断条件后，最少 12 比特时间内，发送器将 TXD 线变回 1。因此，发送器保证远程接收器正确检测到间断结束并开始下一个字符。若时间保障值大于 12，TXD 线将在时间保障周期内保持高电平。

将 TXD 线保持这一周期后，发送器恢复正常工作。

Figure 115 给出 TXD 线上开始间断 (STTBRK) 与停止间断 (STP BRK) 命令的作用。

Figure 115. 间断传输



接收间断

当所有数据、检验及停止位均为低时，接收器检测到间断条件。这与当数据为 0x00 但 FRAME 为低时的检测结果相当。

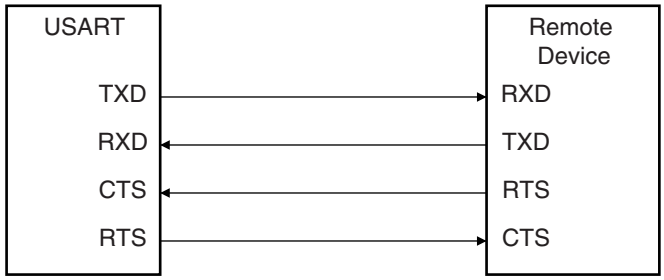
当检测到低电平的停止位时，接收器将US_CSR寄存器的RXBRK位使能，该位可通过在US_CR寄存器的 RSTSTA 位写 1 来清零。

异步工作模式下检测到至少 2/16 的比特周期为高电平，或在同步工作模式下有一个采样值为高，表明接收间断结束。间断结束也可通过 RXBRK 置位来实现。

硬件握手

USART 有硬件握手段外流控制特性。RTS 与 CTS 引脚与远程器件连接，如 Figure 116 所示。

Figure 116. 与远程器件连接的硬件握手



在 US_MR 寄存器 USART_MODE 域写入 0x2，则 USART 将执行硬件握手操作。

除接收器对 RTS 引脚及发送器对 CTS 引脚电平改变外，硬件握手使能后的 USART 操作与标准同步或异步模式下相同 。使用该模式需要用 PDC 通道接收数据。发送器在任何情况下均可处理硬件握手。

Figure 117给出当硬件握手使能时接收器的操作。若接收器禁用且来自PDC通道的RXBUFF (接收缓冲满) 状态为高， RTS 引脚拉高。通常当 CTS 引脚 (由 RTS 驱动) 为高时远程器件不会开始发送。一旦接收器使能， RTS 变低，则表示远程器件可启动发送。给 PDC 定义一个新缓冲器，将 RXBUFF 状态位清零，则 RTS 引脚随之变低。

Figure 117. 硬件握手时接收器工作情况

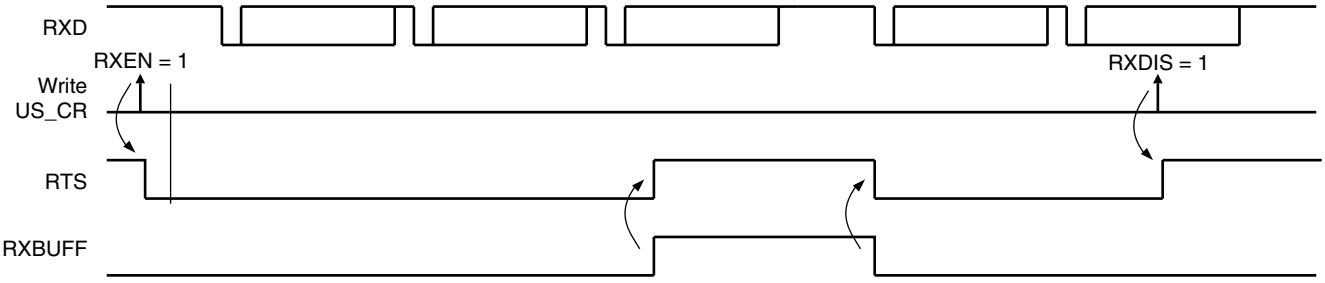
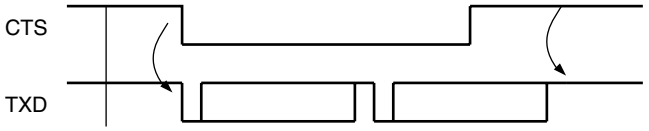


Figure 118 给出硬件握手使能后，发送器如何操作。CTS 引脚禁用发送器。若有字符正在处理，则在当前字符处理完成后将发送器禁用，一旦 CTS 变低即开始下一个字符的发送。

Figure 118. 硬件握手时发送器工作情况



ISO7816 模式

USART有一个与ISO7816兼容模式。该模式允许与智能卡连接并可通过ISO7816链接与安全访问模块 (SAM) 通信。支持 ISO7816 规范定义的 T = 0 与 T = 1 协议。

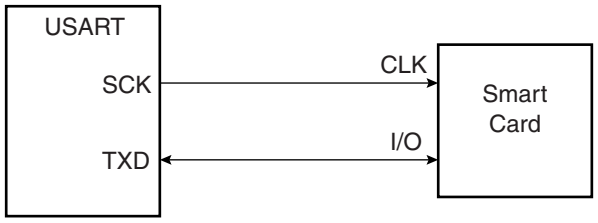
将 US_MR 寄存器 USART_MODE 域写 0x4，将设置 USART 在 ISO7816 的 T = 0 模式下工作；若 USART_MODE 域写 0x5，则 USART 在 ISO7816 的 T = 1 模式下工作。

ISO7816 模式概述

ISO7816为一条双向线的半双工通信。波特率由远程器件提供的时钟分频提供(见“波特率发生器” on page 274)。

USART与智能卡的连接见 Figure 119。TXD线变为双向，波特率发生器由SCK引脚向ISO7816提供时钟。由于 TXD 引脚变为双向，其输出由发送器输出驱动，但只有当发送器激活时其输入为接收器输入。由于 USART 产生时钟，因此它被视为通信主机。

Figure 119. 智能卡与 USART 的连接



无论工作在 ISO7816 的 T = 0 或 T = 1 模式下，字符格式固定。不管 CHRL、 MODE9、 PAR 及 CHMODE 域中是什么值，其配置为 8 位数据位，偶检验及 1 或 2 位停止位。 MSBF 可设置发送是高位在先还是低位在先。

由于通信不是双向的，USART 不能同时在发送器与接收器模式下操作。因而必须根据需要使能或禁用接收器或发送器。 ISO7816 模式下同时使能发送器与接收器结果无法预知。

ISO7816 规范定义了一个反转发送格式。字符的数据位必须在 I/O 线上以其负值发送。USART 不支持该格式，因此在将其写入发送保持寄存器 (US_THR) 前或由接收保持寄存器 (US_RHR) 读出后必须进行一个额外的或操作。

T = 0 协议

T = 0 协议中，字符由一个起始位、八个数据位、一个奇偶校验位及一个两比特时间的保障时间组成。在保障时间中，发送器移出比特但不驱动 I/O 线。

若无检验错误检测，保障时间内 I/O 线保持为 1，且发送器可继续发送下个字符，见 Figure 120。

若接收器检测到检验错误，在保障时间内它将 I/O 线驱动为 0，如 Figure 121 所示。该错误位也称为 NACK，即无应答。此时，由于保障时间长度未变而增加了一个 1 比特时间的错误位时间，因此字符加长 1 比特时间。

当 USART 为接收器且检测到错误，它不会将字符载入接收保持寄存器 (US_RHR)。而是适当设置状态寄存器 (US_SR) 的 PARE 位以使软件处理该错误。

Figure 120. 没有检验错误的 T = 0 协议

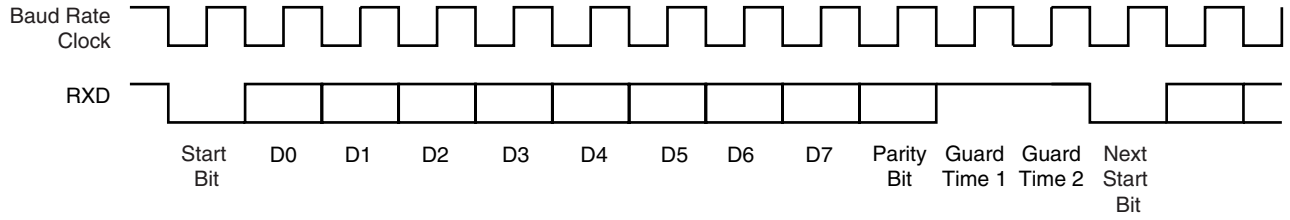
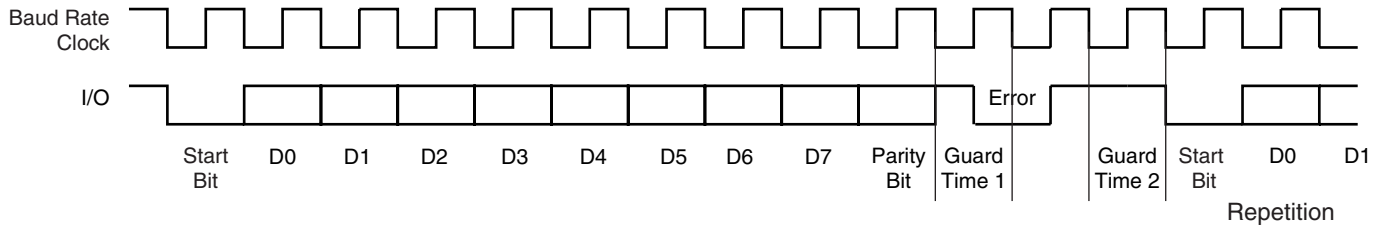


Figure 121. 有检验错误的 T = 0 协议



接收错误计数器

USART 接收器还记录错误总数。可从错误数目寄存器 (US_NER) 中读出。NB_ERRORS 域可记录 255 个错误。读取 US_NER 自动清除 NB_ERRORS 域。

接收 NACK 抑制

USART 可配置为抑制错误。通过置位 US_MR 的 INACK 位实现。若 INACK 为 1，即使检测到检验位，I/O 线上没有错误信号，但 US_SR 寄存器中 INACK 位置位。可通过写控制寄存器 (US_CR) 的 RSTNACK 位为 1 来清除 INACK 位。

此外，若 INACK 置位，接收到的错误字符保存在接收保持寄存器中，就像没有错误出现。但 RXRDY 位不会升高。

发送字符复制

当 USART 正在发送字符并得到 NACK 时，在移到下一个字符前，它可自动复制该字符。通过在 US_MR 寄存器 MAX_ITERATION 域中写入大于 0 的值使能复制。每个字符最多可发送八次；第一次发送加七次复制发送。

若 MAX_ITERATION 不等于 0， USART 复制字符的次数与 MAX_ITERATION 中值相同。

当 USART 复制次数达到 MAX_ITERATION 值时，通道状态寄存器 (US_CSR) 中 ITERATION 位置位。若复制字符得到接收器应答，复制停止并将迭代计数器清零。

US_CSR 寄存器中 ITERATION 位可通过在控制寄存器的 RSIT 位写 1 清零。

禁用连续接收 NACK

接收器可限制返回远程发送器的连续 NACK。通过设置 US_MR 寄存器的 DSNACK 位实现。NACK发送的最多数目在MAX_ITERATION域中编程。一旦达到MAX_ITERATION，认为字符正确，线上发送应答并将通道状态寄存器的 ITERATION 位置位。

T = 1 协议

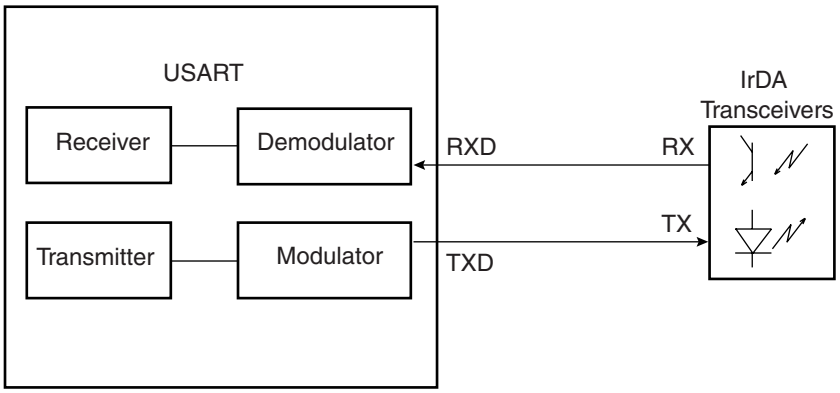
当工作在 ISO7816 的 T = 1 协议下，发送操作与只有一位停止位的异步格式相似。当发送时产生奇偶校验位，在接收时对其检测。奇偶错误检测通过设置通道状态寄存器 (US_CSR) 的 PARE 位实现。

IrDA 模式

USART的IrDA模式支持半双工点对点无线通信。它内置了与红外收发器无连接的调制器和解调器，见 Figure 122。调制器与解调器适用于 IrDA 规范版本 1.1，支持的数据传输速度范围从 2.4 Kb/s 到 115.2 Kb/s。

在 US_MR 寄存器的 USART_MODE 域写 0x8 将使能 USART IrDA 模式。 IrDA 滤波寄存器 (US_IF) 可配置解调滤波器。USART发送器与接收器工作在正常异步模式下，所有参数可访问。注意，调制器与解调器均处于激活状态。

Figure 122. 与 IrDA 收发器连接



接收器与发送器必须根据传输方向使能或禁用。

IrDA 调制

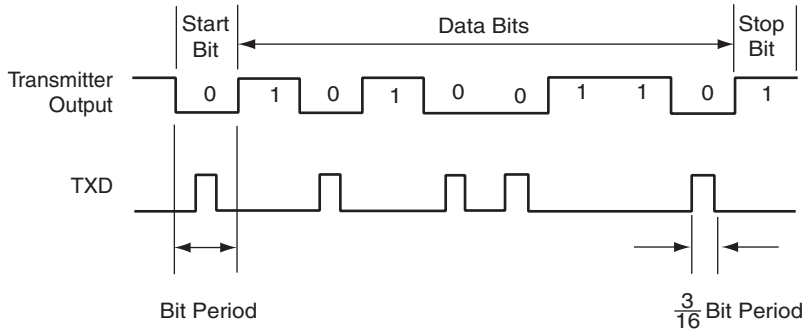
当波特率小于或等于 115.2 Kb/s，使用 RZI 调制方案。由一个 3/16 比特周期的轻脉冲表示 "0"。Table 66 中给出一些信号脉冲持续时间。

Table 66. IrDA 脉冲持续时间

波特率	脉冲持续时间 (3/16)
2.4 Kb/s	78.13 µs
9.6 Kb/s	19.53 µs
19.2 Kb/s	9.77 µs
38.4 Kb/s	4.88 µs
57.6 Kb/s	3.26 µs
115.2 Kb/s	1.63 µs

Figure 123 给出字符发送的示例。

Figure 123. IrDA 调制



IrDA 波特率

Table 67 给出一些 CD 值、波特率误差及脉冲持续时间的例子。注意，可接受的最高误差为 ±1.87%。

Table 67. IrDA 波特率误差

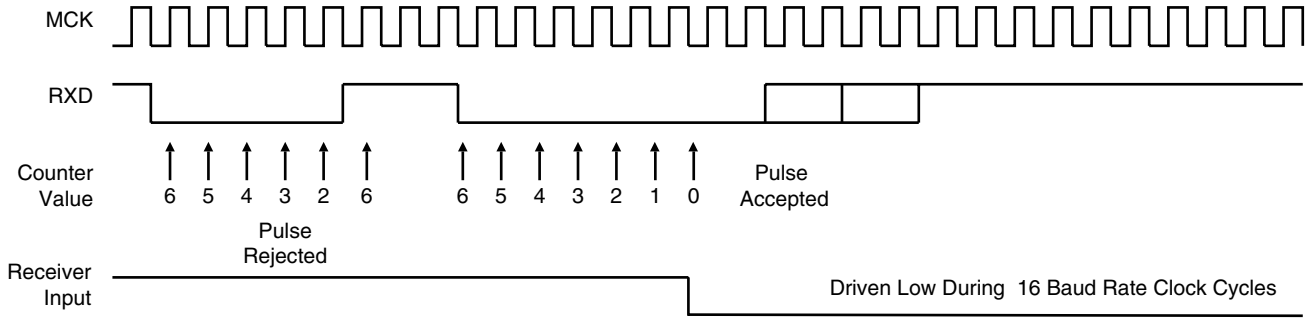
外设时钟	波特率	CD	波特率误差	脉冲时间
3 686 400	115 200	2	0.00%	1.63
20 000 000	115 200	11	1.38%	1.63
32 768 000	115 200	18	1.25%	1.63
40 000 000	115 200	22	1.38%	1.63
3 686 400	57 600	4	0.00%	3.26
20 000 000	57 600	22	1.38%	3.26
32 768 000	57 600	36	1.25%	3.26
40 000 000	57 600	43	0.93%	3.26
3 686 400	38 400	6	0.00%	4.88
20 000 000	38 400	33	1.38%	4.88
32 768 000	38 400	53	0.63%	4.88
40 000 000	38 400	65	0.16%	4.88
3 686 400	19 200	12	0.00%	9.77
20 000 000	19 200	65	0.16%	9.77
32 768 000	19 200	107	0.31%	9.77
40 000 000	19 200	130	0.16%	9.77
3 686 400	9 600	24	0.00%	19.53
20 000 000	9 600	130	0.16%	19.53
32 768 000	9 600	213	0.16%	19.53
40 000 000	9 600	260	0.16%	19.53
3 686 400	2 400	96	0.00%	78.13
20 000 000	2 400	521	0.03%	78.13
32 768 000	2 400	853	0.04%	78.13

IrDA 解调器

解调器基于 IrDA 接收滤波器，包含一个由 US_IF 载入值的 8 位向下计数器。当检测到 RXD 引脚下降沿，滤波计数器以主机时钟 (MCK) 速度向下计数。若检测到 RXD 引脚上升沿，计数器停止并重新载入 US_IF 值。若计数器到达 0 仍未检测到上升沿，在一个比特时间内计数器输入拉低。

Figure 124 给出 IrDA 解调器的操作。

Figure 124. IrDA 解调器操作

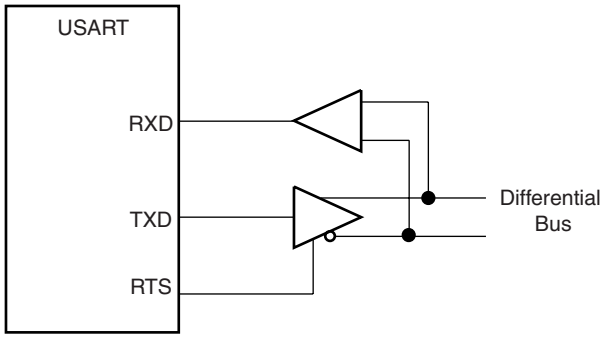


由于 IrDA 模式与 ISO7816 使用相同的逻辑，因此要注意 US_FIDI 中的 FI_DI_RATIO 域值必须大于 0，以确保 IrDA 通信操作正确。

RS485 模式

USART 的 RS485 模式使能线驱动控制。在 RS485 模式下 ,USART 与异步或同步模式下操作相同，并可对所有参数进行配置。不同处在于当发送器工作时将 RTS 引脚拉高。 RTS 引脚动作由 TXEMPTY 位控制。Figure 125 给出了 USART 与 RS485 总线的典型连接。

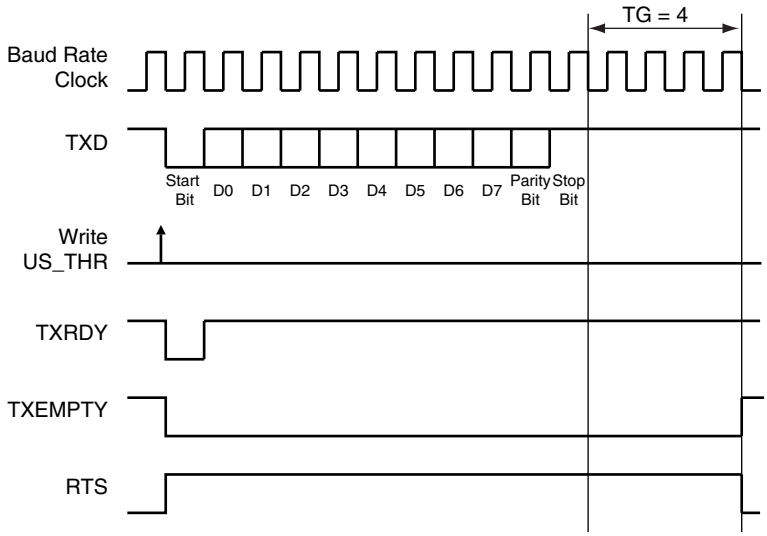
Figure 125. 与 RS485 总线的典型连接



在 US_MR 寄存器的 USART_MODE 域写入 0x1，可将 USART 设置为 RS485 模式。

RTS 引脚电平与 TXEMPTY 位电平相反。注意 ,当时间保障编程时 RTS 引脚为高 ,因此在最后一个字符完成后线依然为高。 Figure 126 给出当时间保障使能，发送字符时的 RTS 波形。

Figure 126. 有时间保障的 RTS 驱动示例



调制解调模式

USART 调制解调模式使能下列信号控制:DTR (数据终端就绪)、DSR (时间设置就绪)、RTS (发送请求)、CTS (发送清零)、DCD (数据载波检测) 及 RI (环指示器)。在调制解调模式下，USART 作为 DTE (数据终端设备) 使用 ,驱动 DTR 与 RTS ,并可检测 DSR、DCD、CTS 及 RI 上电平变化。

在 US_MR 寄存器 USART_MODE 域写入 0x3，可将 USART 设置为调制解调模式。 USART 工作在调制解调模式模式下操作与异步模式下相同，可对所有参数进行配置。

Table 68 给出调制解调连接模式下对应的 USART 信号。

Table 68. 电路参考

USART 引脚	V24	CCITT	方向
TXD	2	103	由终端到调制解调器
RTS	4	105	由终端到调制解调器
DTR	20	108.2	由终端到调制解调器
RXD	3	104	由调制解调器到终端
CTS	5	106	由终端到调制解调器
DSR	6	107	由终端到调制解调器
DCD	8	109	由终端到调制解调器
RI	22	125	由终端到调制解调器

在控制寄存器 (US_CR) 的 RTSDIS、 RTSEN、 DTRDIS 与 DTREN 位分别置 1，可控制 RTS 与 DTR 输出引脚。禁用命令强制相应位为无效电平，即高电平；使能命令强制相应位为有效电平，即低电平。

对 RI、DSR、DCD 即 CTS 引脚进行电平检测。若检测到输入变化 ,将通道状态寄存器 (US_CSR) 中对应的 RIIC、 DSRIC、 DCDIC 及 CTSIC 位置位并可触发中断。当读取 US_CSR 时，状态自动清除。此外，当检测到处于无效状态时， CTS 自动禁用发送器。若 CTS 变高时字符正在发送，则字符发送完成后，发送器才真正禁用。

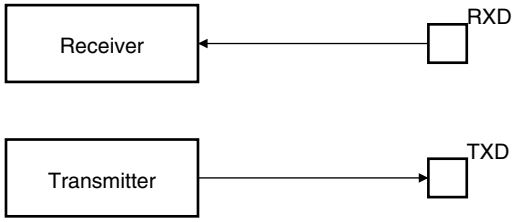
测试模式

USART 有三种不同的测试模式。内部回环可实现板上诊断。回环模式下，USART 接口引脚可不连接，并重新配置为内部或外部回环。

普通模式

普通模式下将 RXD 引脚与接收器输入连接，而发送器输出与 TXD 引脚连接。

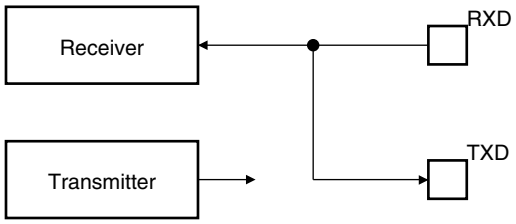
Figure 127. 普通模式配置



自动回应模式

自动回应模式允许位重发。当 RXD 引脚收到一位，它发送到 TXD 引脚，见 Figure 128。对发送器编程不影响 TXD 引脚。 RXD 引脚仍与接收器输入连接，因此接收器保持激活状态。

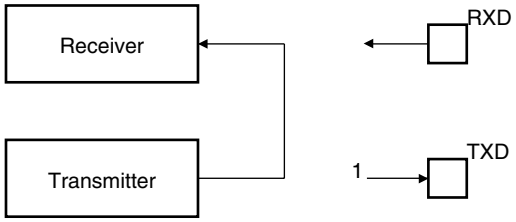
Figure 128. 自动回应模式配置



本地回环模式

本地回环模式下，发送器输出直接与接收器输入连接，如 Figure 129 所示。 TXD 与 RXD 引脚未使用。 RXD 引脚对接收器无效，而 TXD 引脚如空闲状态一样，始终为高。

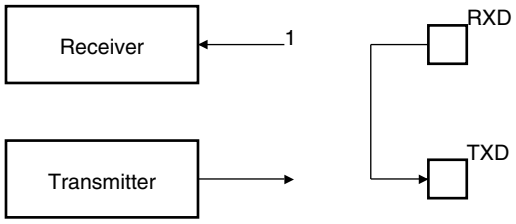
Figure 129. 本地回环模式配置



远程回环模式

远程回环模式下，直接将 RXD 引脚与 TXD 引脚连接，如 Figure 130 所示。发送器与接收器禁用无效。该模式允许位重发。

Figure 130. 远程回环模式配置



USART 用户接口

Table 69. USART 寄存器映射

偏移	寄存器	名称	访问类型	复位状态
0x0000	控制寄存器	US_CR	只写	–
0x0004	模式寄存器	US_MR	读 / 写	–
0x0008	中断使能寄存器	US_IER	只写	–
0x000C	中断禁用寄存器	US_IDR	只写	–
0x0010	中断屏蔽寄存器	US_IMR	只读	0
0x0014	通道状态寄存器	US_CSR	只读	–
0x0018	接收器保持寄存器	US_RHR	只读	0
0x001C	发送器保持寄存器	US_THR	只写	–
0x0020	波特率发生器寄存器	US_BRGR	读 / 写	0
0x0024	接收器超时寄存器	US_RTOR	读 / 写	0
0x0028	发送器时间保障寄存器	US_TTGR	读 / 写	0
0x002C - 0x003C	保留	–	–	–
0x0040	FI DI 比率寄存器	US_FIDI	读 / 写	0x174
0x0044	错误数目寄存器	US_NER	只读	–
0x0048	保留	–	–	–
0x004C	IrDA 滤波寄存器	US_IF	读 / 写	0
0x0050 - 0x00FC	保留	–	–	–
0x0100 - 0x0128	保留给 PDC 寄存器	–	–	–

USART 控制寄存器

寄存器名称 : US_CR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	RTSDIS	RTSEN	DTRDIS	DTREN

15	14	13	12	11	10	9	8
RETTO	RSTNACK	RSTIT	SENDA	STTTO	STPBRK	STTBRK	RSTSTA

7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	–

- **RSTRX: 接收器复位**

0：无效。

1：接收器复位。

- **RSTTX: 发送器复位**

0：无效。

1：发送器复位。

- **RXEN: 接收器使能**

0：无效。

1：若 RXDIS 为 0，使能接收器。

- **RXDIS: 接收器禁用**

0：无效。

1：禁用接收器。

- **TXEN: 发送器使能**

0：无效。

1：若 TXDIS 为 0，使能发送器。

- **TXDIS: 发送器禁用**

0：无效。

1：禁用发送器。

- **RSTSTA: 状态位复位**

0：无效。

1：US_CSR 寄存器中状态位 PARE、 FRAME、 OVRE 与 RXBRK 复位。

- **STTBRK: 启动间断**

0：无效。

1：在 US_THR 中有字符且发送移位寄存器中字符已发送后，开始发送间断。若间断已发送则无效。

- **STPBRK: 停止间断**

0：无效。

1：最少一字符时间且发送高电平达 12 比特周期后停止间断发送。若间断已发送则无效。

- **STTTO: 启动超时**

0：无效。

1：在超时计数器计数前等待一个字符。

- **SENDA: 发送地址**

0：无效。

1：只适用于多点模式，发送写入 US_THR 的下一个字符并将地址位置位。

- **RSTIT: 迭代复位**

0：无效。

1：US_CSR 寄存器中 ITERATION 复位，若没有使能 ISO7816 则无效。

- **RSTNACK: 无应答复位**

0：无效。

1：US_CSR 寄存器中 NACK 复位。

- **RETTO: 重新超时**

0：无效。

1：超时重启。

- **DTREN: 数据终端就绪使能**

0：无效。

1：将 DTR 引脚驱动为 0。

- **DTRDIS: 数据终端就绪禁用**

0：无效。

1：将 DTR 引脚驱动为 1。

- **RTSEN: 发送请求使能**

0：无效。

1：将 RTS 引脚驱动为 0。

- **RTSDIS: 发送请求禁用**

0：无效。

1：将 RTS 引脚驱动为 1。

USART 模式寄存器

寄存器名称 : US_MR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	FILTER	–	MAX_ITERATION		

23	22	21	20	19	18	17	16
–	–	DSNACK	INACK	OVER	CLKO	MODE9	MSBF

15	14	13	12	11	10	9	8
CHMODE		NBSTOP		PAR			SYNC

7	6	5	4	3	2	1	0
CHRL		USCLKS		USART_MODE			

- **USART_MODE**

USART_MODE				USART 模式
0	0	0	0	普通
0	0	0	1	RS485
0	0	1	0	硬件握手
0	0	1	1	调制解调
0	1	0	0	IS07816 协议：T = 0
0	1	0	1	保留
0	1	1	0	IS07816 协议：T = 1
0	1	1	1	保留
1	0	0	0	IrDA
1	1	x	x	保留

- **USCLKS: 时钟选择**

USCLKS		时钟选择
0	0	MCK
0	1	MCK / DIV
1	0	保留
1	1	SCK

- **CHRL: 字符长度 .**

CHRL		字符长度
0	0	5 位
0	1	6 位
1	0	7 位
1	1	8 位

- **SYNC: 同步模式选择**

0：USART 工作在异步模式下。

1：USART 工作在同步模式下。

- **PAR: 检验类型**

PAR			检验类型
0	0	0	偶检验
0	0	1	奇检验
0	1	0	检验强制为 0 (空)
0	1	1	检验强制为 1 (标志)
1	0	x	无检验
1	1	x	多点模式

- **NBSTOP: 停止位数**

NBSTOP		异步 (SYNC = 0)	同步 (SYNC = 1)
0	0	1 个停止位	1 个停止位
0	1	1.5 个停止位	保留
1	0	2 个停止位	2 个停止位
1	1	保留	保留

- **CHMODE: 通道模式**

CHMODE		模式说明
0	0	普通模式
0	1	自动回应。接收器输入与 TXD 引脚连接。
1	0	本地回环。发送器输出与接收器输入连接。
1	1	远程回环。 RXD 引脚与 TXD 引脚连接。

- **MSBF: 位次序**

0：低位在先。

1：高位在先。

- **MODE9: 9 位字符长度**

0：CHRL 定义字符长度。

1：9 位字符长度。

- **CKLO: 时钟输出选择**

0：USART 不驱动 SCK 引脚。

1：若 USCLKS 未选择外部时钟 SCK ， USART 驱动 SCK 引脚。

- **OVER: 重采样模式**

0：16 倍重采样。

1：8 倍重采样。

- **INACK: 抑制无应答**

0：产生 NACK。

1：不产生 NACK。

- **DSNACK: 连续 NACK 禁用**

0：一旦收到的字符出现检验错误， NACK 发送到 ISO 线上 (除非 INACK 置位)。

1:连续检验错误达到 MAX_ITERATION 域给出的值。这些检验错误在 ISO 线上产生 NACK。一旦达到该值 ,ISO 线上不再发送另外的 NACK。出现 ITERATION 标志。

- **MAX_ITERATION**

定义模式 ISO7816，协议 T= 0 下最大迭代数。

- **FILTER: 红外接收线滤波器**

0：USART 不对接收线滤波。

1：USART 使用 3 采样滤波器 (1/16 位时钟) (2/3 多) 对接收线滤波。

USART 中断使能寄存器

寄存器名称： US_IER

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	CTSIC	DCDIC	DSRIC	RIIC

15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT

7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: RXRDY 中断使能**
- **TXRDY: TXRDY 中断使能**
- **RXBRK: 接收器间断中断使能**
- **ENDRX: 接收结束中断使能**
- **ENDTX: 发送结束中断使能**
- **OVRE: 溢出错误中断使能**
- **FRAME: 帧错误中断使能**
- **PARE: 奇偶错误中断使能**
- **TIMEOUT: 超时中断使能**
- **TXEMPTY: TXEMPTY 中断使能**
- **ITERATION: 迭代中断使能**
- **TXBUFE: 缓冲器空中断使能**
- **RXBUFF: 缓冲器满中断使能**
- **NACK: 无应答中断使能**
- **RIIC: 环指示器输入变化使能**
- **DSRIC: 数据设置就绪输入变化使能**
- **DCDIC: 数据载波检测输入变化中断使能**
- **CTSIC: 发送输入变化清除中断使能**

0：无效。

1：使能相应中断。

USART 中断禁用寄存器

寄存器名称： US_IDR

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	CTSIC	DCDIC	DSRIC	RIIC

15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT

7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: RXRDY 中断禁用**
- **TXRDY: TXRDY 中断禁用**
- **RXBRK: 接收器间断中断禁用**
- **ENDRX: 接收结束中断禁用**
- **ENDTX: 发送结束中断禁用**
- **OVRE: 溢出错误中断禁用**
- **FRAME: 帧错误中断禁用**
- **PARE: 奇偶错误中断禁用**
- **TIMEOUT: 超时中断禁用**
- **TXEMPTY: TXEMPTY 中断禁用**
- **ITERATION: 迭代中断禁用**
- **TXBUFE: 缓冲器空中断禁用**
- **RXBUFF: 缓冲器满中断禁用**
- **NACK: 无应答中断禁用**
- **RIIC: 环指示器输入变化禁用**
- **DSRIC: 数据设置就绪输入变化禁用**
- **DCDIC: 数据载波检测输入变化中断禁用**
- **CTSIC: 发送输入变化清除中断禁用**

0：无效。

1：禁用相应中断。

USART 中断屏蔽寄存器

寄存器名称： US_IMR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	CTSIC	DCDIC	DSRIC	RIIC

15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT

7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: RXRDY 中断屏蔽**
- **TXRDY: TXRDY 中断屏蔽**
- **RXBRK: 接收器间断中断屏蔽**
- **ENDRX: 接收结束中断屏蔽**
- **ENDTX: 发送结束中断屏蔽**
- **OVRE: 溢出错误中断屏蔽**
- **FRAME: 帧错误中断屏蔽**
- **PARE: 奇偶错误中断屏蔽**
- **TIMEOUT: 超时中断屏蔽**
- **TXEMPTY: TXEMPTY 中断屏蔽**
- **ITERATION: 迭代中断屏蔽**
- **TXBUFE: 缓冲器空中断屏蔽**
- **RXBUFF: 缓冲器满中断屏蔽**
- **NACK: 无应答中断屏蔽**
- **RIIC: 环指示器输入变化屏蔽**
- **DSRIC: 数据设置就绪输入变化屏蔽**
- **DCDIC: 数据载波检测输入变化中断屏蔽**
- **CTSIC: 发送输入变化清除中断屏蔽**

0：相应中断禁用。

1：相应中断使能。

USART 通道状态寄存器

寄存器名称： US_CSR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
CTS	DCD	DSR	RI	CTSIC	DCDIC	DSRIC	RIIC

15	14	13	12	11	10	9	8
–	–	NACK	RXBUFF	TXBUFE	ITERATION	TXEMPTY	TIMEOUT

7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	TXRDY	RXRDY

- **RXRDY: 接收器就绪**

0 : 上次读 US_RHR 后未接收到完整的字符或接收器禁用。若接收器禁用时正在接收字符，当接收器使能时 RXRDY 变为 1。

1 : 上次读 US_RHR 后至少收到一个完整的字符，且 US_RHR 未被读取。

- **TXRDY: 发送器就绪**

0 : US_THR 中字符等待发送到发送移位寄存器，或请求 STTBRK 命令，或发送器禁用。一旦发送器使能，TXRDY 变为 1。

1 : US_THR 中无字符。

- **RXBRK: 间断接收 / 间断结束**

0 : 上次 RSTSTA 后未检测到间断接收或间断结束。

1 : 上次 RSTSTA 后检测到间断接收或间断结束。

- **ENDRX: 接收器传输结束**

0 : 接收 PDC 通道的传输结束信号无效。

1 : 接收 PDC 通道的传输结束信号有效。

- **ENDTX: 发送器传输结束**

0 : 发送 PDC 通道的传输结束信号无效。

1 : 发送 PDC 通道的传输结束信号有效。

- **OVRE: 溢出错误**

0 : 上次 RSTSTA 后未出现溢出错误。

1 : 上次 RSTSTA 后至少出现一次溢出错误。

- **FRAME: 帧错误**

0 : 上次 RSTSTA 后未检测到停止位。

1 : 上次 RSTSTA 后至少检测到一个停止位。

- **PARE: 检验错误**

0 : 上次 RSTSTA 后未检测到检验错误。

1 : 上次 RSTSTA 后至少检测到一次检验错误。

- **TIMEOUT: 接收器超时**

0 : 上次启动超时命令后无超时或超时寄存器为 0。

1 : 上次启动超时命令后有超时。

- **TXEMPTY: 发送器空**

0：US_THR 与发送移位寄存器中均无字符，或发送器禁用。

1：US_THR 与发送移位寄存器中至少有一个字符。

- **ITERATION: 达到最大重复数**

0：上次 RSIT 后还未达到最大重复数。

1：上次 RSIT 后已达到最大重复数。

- **TXBUFE: 发送缓冲器空**

0：发送 PDC 通道缓冲器空信号无效。

1：发送 PDC 通道缓冲器空信号有效。

- **RXBUFF: 接收缓冲器满**

0：接收 PDC 通道缓冲器满信号无效。

1：接收 PDC 通道缓冲器满信号有效。

- **NACK: 无应答**

0：上次 RSTNACK 后未检测到无应答。

1：上次 RSTNACK 后至少检测到一次无应答。

- **RIIC: 环指示器输入变化标志**

0：上次读 US_CSR 后在 RI 引脚上未检测到输入变化。

1：上次读 US_CSR 后在 RI 引脚上至少检测到一次输入变化。

- **DSRIC: 数据设置就绪输入变化标志**

0：上次读 US_CSR 后在 DSR 引脚上未检测到输入变化。

1：上次读 US_CSR 后在 DSR 引脚上至少检测到一次输入变化。

- **DCDIC: 数据载波检测输入变化标志**

0：上次读 US_CSR 后在 DCD 引脚上未检测到输入变化。

1：上次读 US_CSR 后在 DCD 引脚上至少检测到一次输入变化。

- **CTSIC: 发送输入变化清除标志**

0：上次读 US_CSR 后在 CTS 引脚上未检测到输入变化。

1：上次读 US_CSR 后在 CTS 引脚上至少检测到一次输入变化。

- **RI: RI 输入映象**

0：RI 为 0。

1：RI 为 1。

- **DSR: DSR 输入映象**

0：DSR 为 0。

1：DSR 为 1。

- **DCD: DCD 输入映象**

0：DCD 为 0。

1：DCD 为 1。

- **CTS: CTS 输入映象**

0：CTS 为 0。

1：CTS 为 1。

USART 接收保持寄存器

寄存器名称 : US_RHR

访问类型 : 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	RXCHR

7	6	5	4	3	2	1	0
RXCHR							

- **RXCHR: 收到的字符**

若 RXRDY 置位，为接收到的最后一个字符。

USART 发送保持寄存器

寄存器名称 : US_THR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	TXCHR

7	6	5	4	3	2	1	0
TXCHR							

- **TXCHR: 将发送的字符**

若 TXRDY 未置位，则为下一个要发送的字符。

USART 波特率发送器寄存器

寄存器名称 : US_BRGR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
CD							

7	6	5	4	3	2	1	0
CD							

- **CD: 时钟分频器**

CD	USART_MODE ≠ ISO7816			USART_MODE = ISO7816
	SYNC = 0		SYNC = 1	
	OVER = 0	OVER = 1		
0	波特率时钟禁用			
1 到 65535	波特率 = 选定时钟 /16/CD	波特率 = 选定时钟 /8/CD	波特率 = 选定时钟 /CD	波特率 = 选定时钟 /CD/FI_DI_RATIO

USART 接收器超时寄存器

寄存器名称： US_RTOR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
TO							

7	6	5	4	3	2	1	0
TO							

- **TO: 超时值**

0：接收器超时禁用。

1 - 65535：接收器超时使能且超时延迟为 TO x 比特周期。

USART 发送器时间保障寄存器

寄存器名称 : US_TTGR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
TG							

- **TG: 时间保障值**

0：发送器时间保障禁用。

1 - 255：发送器时间保障使能且时间保障延迟为 TG x 比特周期。

USART FI DI 比率寄存器

寄存器名称 : US_FIDI

访问类型 : 读 / 写

复位值 :: 0x174

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	FI_DI_RATIO		

7	6	5	4	3	2	1	0
FI_DI_RATIO							

- **FI_DI_RATIO: FI 与 DI 比值**

0：若选择 ISO7816 模式，波特率发生器不产生信号。

1 - 2047：若选择 ISO7816 模式，波特率为 SCK 时钟与 FI_DI_RATIO 相除后的值。

USART 错误数目寄存器

寄存器名称 : US_NER

访问类型 : 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
NB_ERRORS							

- **NB_ERRORS: 错误数目**

ISO7816 传输中错误总数。当被读取时，该寄存器自动清零。

USART IrDA 滤波器寄存器

寄存器名称 : US_IF

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
IRDA_FILTER							

- **IRDA_FILTER: IrDA 滤波器**

设置 IrDA 解调器滤波器。

同步串行控制器 (SSC)

概述

Atmel同步串行控制器(SSC)提供与外部器件的同步通信。它支持许多用于音频及电信应用中常用的串行同步通信协议，如 I2S、短帧同步、长帧同步等。

SSC 包含独立的接收器、发送器及一个时钟分频器。每个发送器及接收器有三个接口：针对数据的 TD/RD 信号、针对时钟的 TK/RK 信号及针对帧同步的 TF/RF 信号。最多可传输 16 个 32 位数据。可编程设定为自动启动或在帧同步信号检测到不同事件时启动。

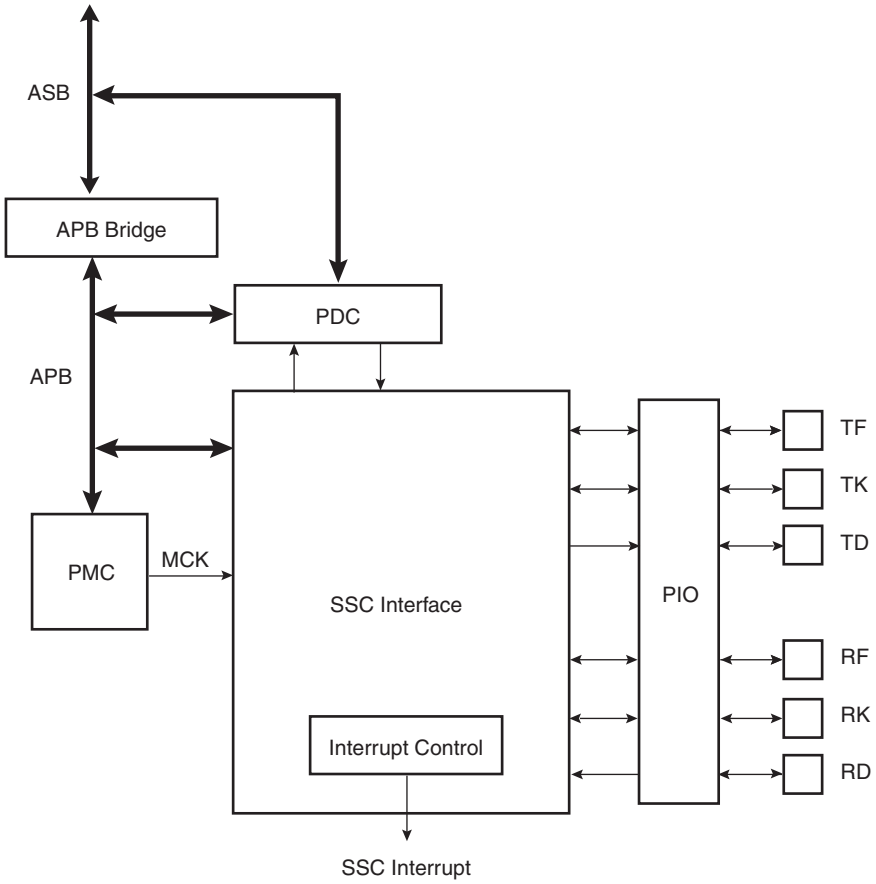
SSC的可编程高电平及两个32位专用PDC 通道，可在没有处理器干涉的情况下进行连续的高速率数据传输。

由于与两个 PDC 通道连接， SSC 可在低处理器开销的情况下与下列器件连接：

- 主机或从机模式下的 CODEC
- 专用串行接口的 DAC，特别是 I2S
- 磁卡阅读器

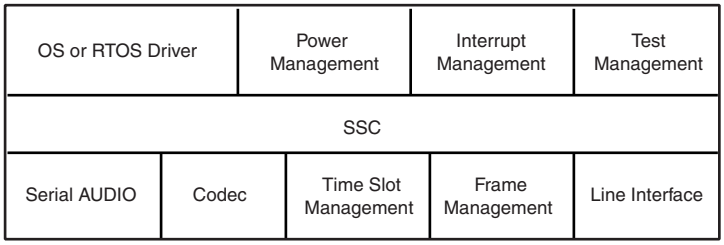
方框图

Figure 131. 方框图



应用框图

Figure 132. 应用框图



引脚名称列表

Table 70. I/O 线说明

引脚名称	引脚说明	类型
RF	接收器帧同步	输入 / 输出
RK	接收器时钟	输入 / 输出
RD	接收器数据	输入
TF	发送器帧同步	输入 / 输出
TK	发送器时钟	输入 / 输出
TD	发送器数据	输出

附属产品

I/O 线

与外设引脚连接可与 PIO 线复用。

使用 SSC 接收器前， PIO 控制器必须将专用的 SSC 接收器 I/O 线配置为 SSC 外设模式。

使用 SSC 发送器前， PIO 控制器必须将专用的 SSC 发送器 I/O 线配置为 SSC 外设模式。

电源管理

SSC时钟不连续。SSC接口可由电源管理控制器(PMC)计时，因此必须先配置PMC以使能SSC时钟。

中断

SSC接口有与高级中断控制器(AIC)连接的中断线。处理中断请求须在配置SSC前对AIC编程。

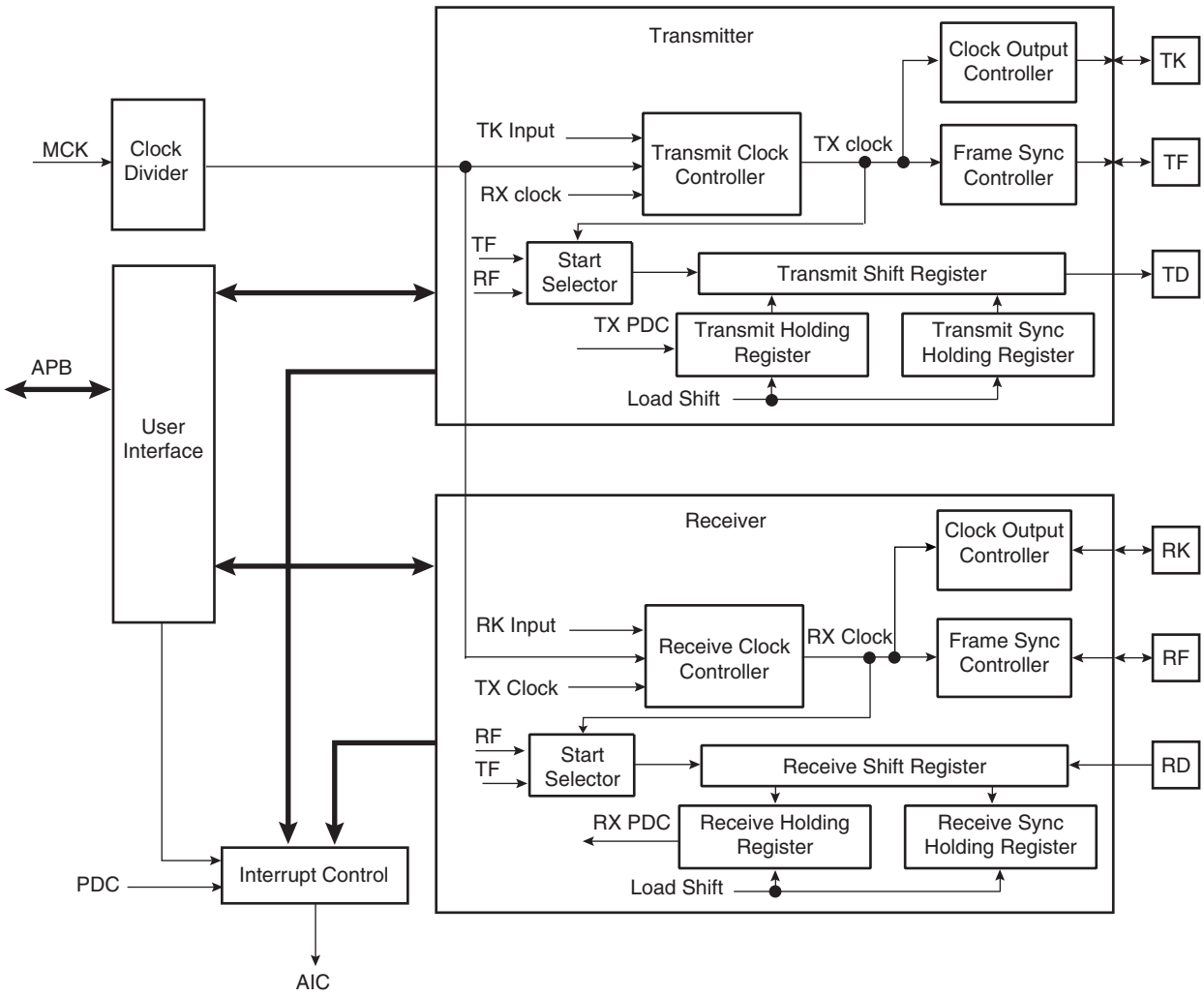
所有 SSC 中断可通过配置 SSC 中断屏蔽寄存器来使能 / 禁用。每个挂起与未屏蔽的 SSC 中断将出现在SSC中断线上。SSC中断服务子程序可通过读取SSC中断状态寄存器来得到中断源。

功能说明

该节包括以下功能说明 : SSC 功能块、时钟管理、数据格式、启动、发送器、接收器及帧同步。

接收器与发送器独立工作。但可通过编程使接收器使用发送时钟或当发送启动时开始数据传输，以使它们同步工作。也可通过编程使发送器使用接收时钟或当接收启动时开始数据传输，实现二者同步工作。发送器与接收器时钟可由 TK 或 RK 引脚提供。使得 SSC 支持多从机模式数据传输。TK 与 RK 引脚的最大时钟速率为主机时钟的 2 分频。每级时钟至少要稳定两个主机时钟。

Figure 133. SSC 功能框图



时钟管理

发送器时钟可由以下产生：

- TK I/O 上接收的外部时钟
- 接收器时钟
- 内部时钟分频器

接收器器时钟可由以下产生：

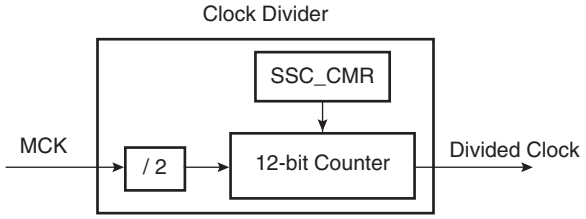
- RK I/O 上接收的外部时钟
- 发送器时钟
- 内部时钟分频器

此外，发送器可在 TK I/O 上产生外部时钟，接收器可在 RK I/O 上产生外部时钟。

因此， SSC 支持多主机与从机模式数据传输。

时钟分频器

Figure 134. 时钟分频框图



主机时钟分频器由时钟模式寄存器SSC_CMR的12位域DIV计数器及比较器(其最大值为4095)确定，最高可对主机时钟 8190 分频。分频后时钟提供给接收器与发送器。当该域编程为 0，时钟分频器不使用并保持无效。

当 DIV 值大于或等于 1，分频后时钟频率为主机时钟 2 倍 DIV 分频后的值。每级分频时钟电平持续时间为主机时钟与 DIV 的乘积。这保证无论 DIV 为奇数还是偶数，分频后时钟占空比为 50% 。

Figure 135. 分频时钟产生

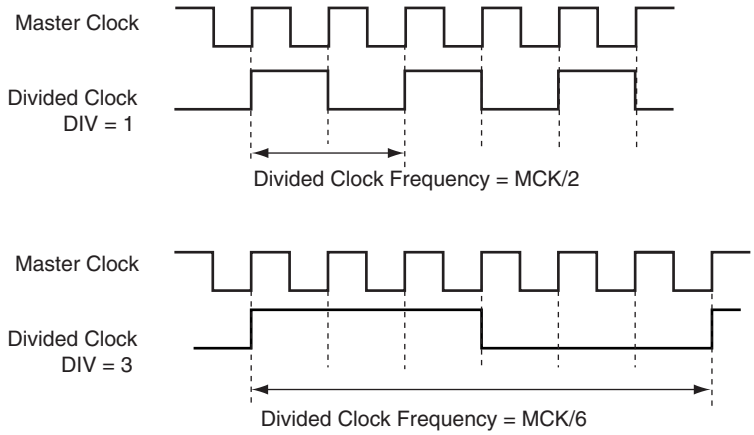


Table 71. 比特率

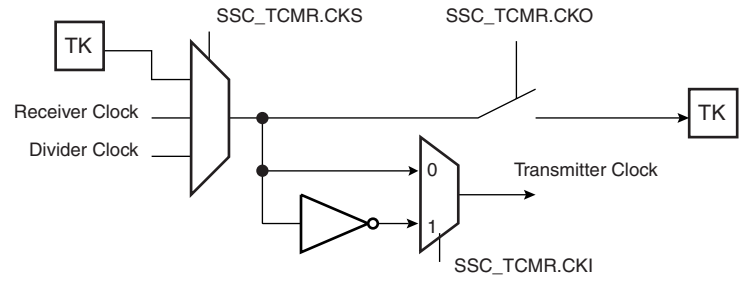
最大	最小
MCK / 2	MCK / 8190

发送器时钟管理

发送器时钟来自接收器时钟或分频器时钟或 TK I/O 上的外部时钟。发送器时钟由 SSC_TCMR (发送时钟模式寄存器) 的 CKS 域选择。发送时钟可通过 SSC_TCMR 的 CKI 位反转。

发送器可连续驱动 TK I/O 或受限于实际数据传输。时钟输出由 SSC_TCMR 寄存器配置。发送时钟反转 (CKI) 位对时钟输出无效。对 TCMR 寄存器编程选择 TK 引脚 (CKS 域) 但同时连续发送时钟 (CKO 域) 可能引起不可预见的结果。

Figure 136. 发送器时钟管理

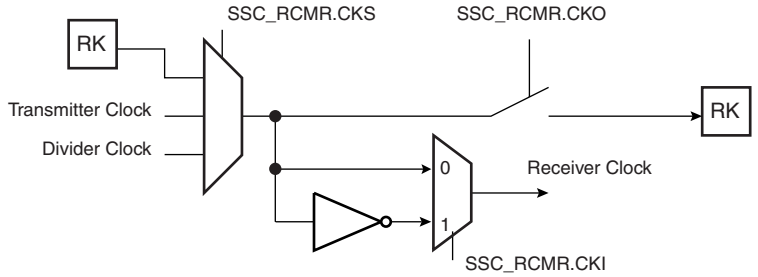


接收器时钟管理

接收器时钟来自发送器时钟或分频器时钟或 RK I/O 上的外部时钟。接收时钟由 SSC_RCMR (接收时钟模式寄存器) 的 CKS 域选择。接收时钟可通过 SSC_RCMR 的 CKI 位反转。

接收器可连续驱动 RK I/O 或受限于实际数据传输。时钟输出由 SSC_RCMR 寄存器配置。接收时钟反转 (CKI) 位对时钟输出无效。对 RCMR 寄存器编程选择 RK 引脚 (CKS 域) 但同时连续接收时钟 (CKO 域) 可能引起不可预见的结果。

Figure 137. 接收器时钟管理



发送器操作

发送帧由启动事件触发，并可在数据发送前加入同步数据。

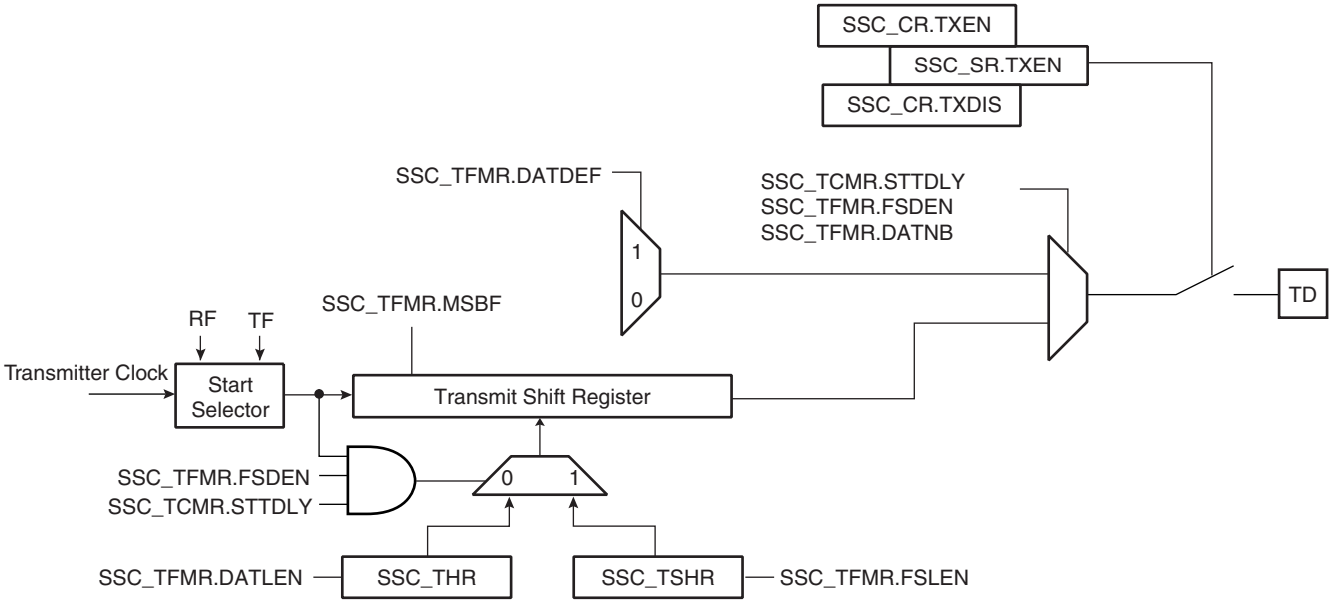
启动事件通过时钟发送时钟模式寄存器 (SSC_TCMR) 来配置，见 See “ 启动 ” on page 322.。

帧同步通过发送帧模式寄存器 (SSC_TFMR) 来配置，见 See “ 帧同步 ” on page 324.。

发送数据时，发送器使用发送器时钟信号作为移位寄存器时钟并在 SSC_TCMR 中选择启动模式。根据应用将数据写入 SSC_THR 寄存器，然后根据选择的数据格式将数据传输到移位寄存器中。

当 SSC_THR 与发送移位寄存器均为空时，SSC_SR 中状态标志 TXEMPTY 置位。当发送保持寄存器传输到移位寄存器时， SSC_SR 中 TXRDY 位置位，新数据可载入保持寄存器中。

Figure 138. 发送器框图



接收器操作

接收器帧由启动事件触发，并可在数据发送前加入同步数据。

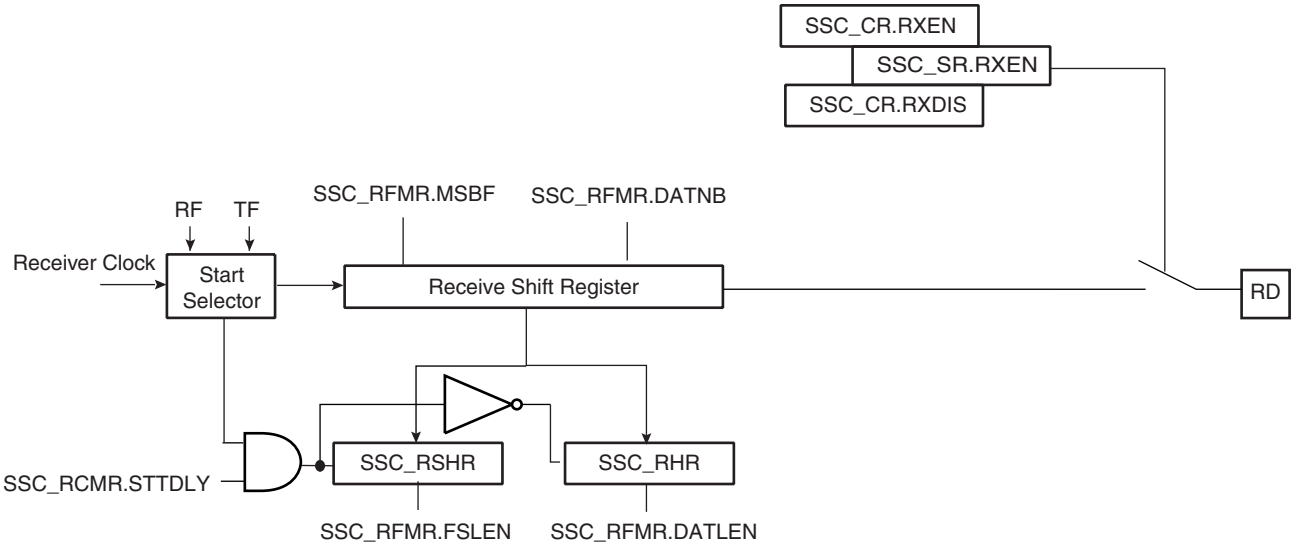
启动事件通过时钟接收时钟模式寄存器 (SSC_RCMR) 来配置，见 See “ 启动 ” on page 322.。

帧同步通过接收帧模式寄存器 (SSC_RFMR) 来配置，见 See “ 帧同步 ” on page 324.。

接收数据时，接收器使用接收器时钟信号作为移位寄存器时钟并在 SSC_RCMR 中选择启动模式。根据选择的数据格式将数据传输到移位寄存器中。

当接收移位寄存器满， SSC 将数据送入保持寄存器， SSC_SR 中状态标志 RXRDY 置位，若在读 RHR 寄存器前有其它传输出现， SSC_SR 中 OVERUN 位置位，且接收器移位寄存器传输到 RHR 寄存器。

Figure 139. 接收器框图



启动

发送器与接收器可编程设定为当事件出现时开始工作，分别设置 SSC_TCMR 中的发送启动选择 (START) 域及 SSC_RCMR 中的接收启动选择 (START) 域。

在以下情况中，启动事件可独立编程：

- 连续。这种情况下，一旦 SSC_THR 中写入数据，即开始发送，并且在接收器时接收启动。
- 与发送器 / 接收器同步
- 检测到 TK/RK 的上升 / 下降沿
- 检测到 TK/RK 的高 / 低电平
- 检测到 TK/RK 的电平变化或跳变沿

可在发送 / 接收时钟寄存器 (RCMR/TCMR) 的任意边沿以同样的方法编程设置。因此，可在 TF (发) 或 RF (接收) 启动。

通过发送 / 接收帧模式寄存器 (TFMR/RFMR) 的 FSOS 域，检测 TF/RF 输入 / 输出。

帧同步必须在与其相关输出上产生。

Figure 140. 发送启动模式

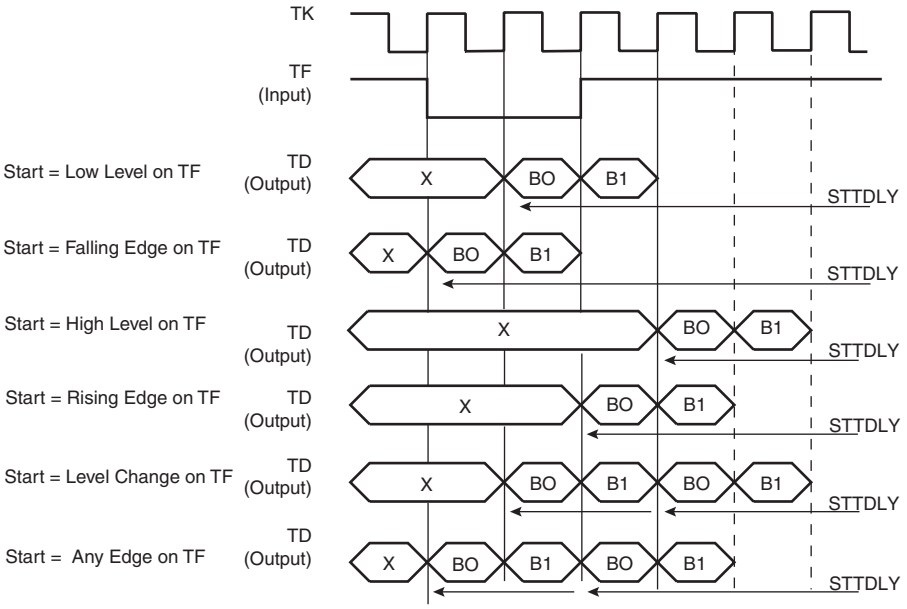
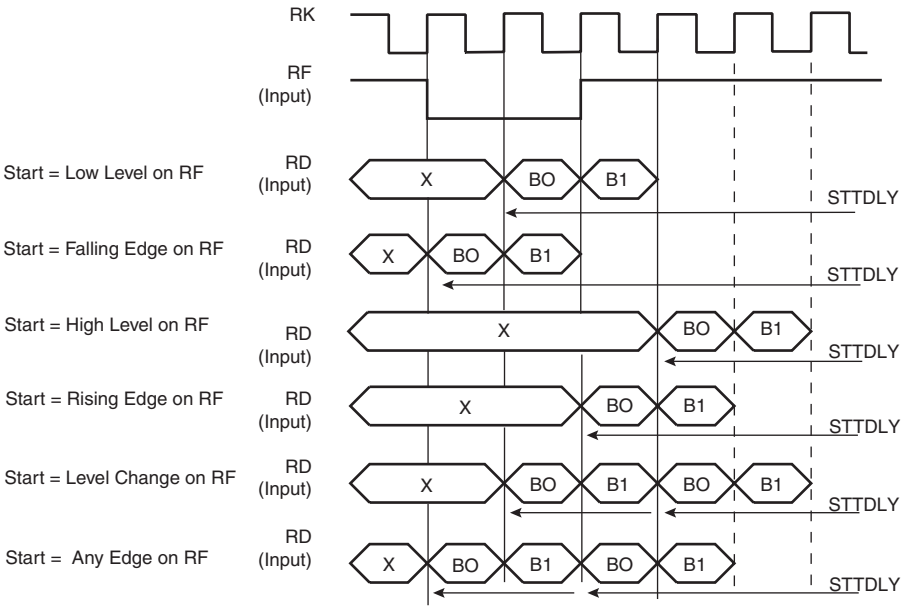


Figure 141. 接收脉冲 / 边沿启动模式



帧同步

发送器与接收器帧同步引脚 TF 与 RF 可编程产生不同类型帧同步信号。发送帧模式寄存器 (SSC_RFMR) 中的及发送帧模式寄存器 (SSC_TFMR) 中的帧同步输出选择 (FSOS) 域用来选择所需波形。

- 支持数据传输中的可编程高低电平。
- 支持数据传输启动前高电平或电平切换。

若选择脉冲波形， SSC_RFMR 与 SSC_TFMR 中帧同步长度 (FSLEN) 域对脉冲长度编程，由 1 比特时间到 16 比特时间。

接收与发送帧同步脉冲周期可通过 SSC_RCMR 与 SSC_TCMR 的周期分频器选择 (PERIOD) 域编程设定。

帧同步数据

帧同步数据在帧同步信号中发送或接收一个特定的标记。

帧同步信号中，接收器可对 RD 线采样并在接收同步保持寄存器中保存数据，发送器可将发送同步保持寄存器发送移位寄存器中。在由 SSC_RFMR/SSC_TFMR 中 FSLEN 域编程设定的帧同步信号中可对数据长度进行采样或移出。

考虑接收帧同步数据操作，若帧同步长度等于或小于启动事件与实际数据接收间的延迟时，通过接收移位寄存器的接收同步保持寄存器执行数据采样。

只有当 SSC_TFMR 寄存器中帧同步数据使能 (FSDEN) 位置位时，执行帧同步操作。若帧同步长度等于或小于启动事件与实际数据发送间的延迟时，普通发送优先且将发送同步保持寄存器中的数据送到发送寄存器中，然后再移出。

帧同步边沿检测

帧同步边沿检测由 SSC_RFMR/SSC_TFMR 中 FSEDGE 域编程设定。将 SSC 中状态寄存器 (SSC_SR) 的 RXSYN/TXSYN 标志设定为帧同步边沿检测 (RF/TF 信号)。

数据格式

发送器与接收器的数据帧格式基本上由发送器帧模式寄存器 (SSC_TFMR) 及接收器帧模式寄存器 (SSC_RFMR) 编程设定。任一情况下，用户可分别选择：

- 启动数据传输事件 (START)。
- 启动时间到首个数据位间位周期延迟数 (STTDLY)。
- 数据长度 (DATLEN)。
- 每次启动事件传输的数据数 (DATNB)。
- 每次启动事件同步传输长度 (FSLEN)。
- 比特意义：高位或低位在先 (MSBF)。

  此外，没有进行数据传输时，发送器可用来发送同步并选择 TD 引脚驱动电平。分别由 SSC_TFMR 中帧同步数据使能 (FSDEN) 位及数据默认值 (DATDEF) 位实现。

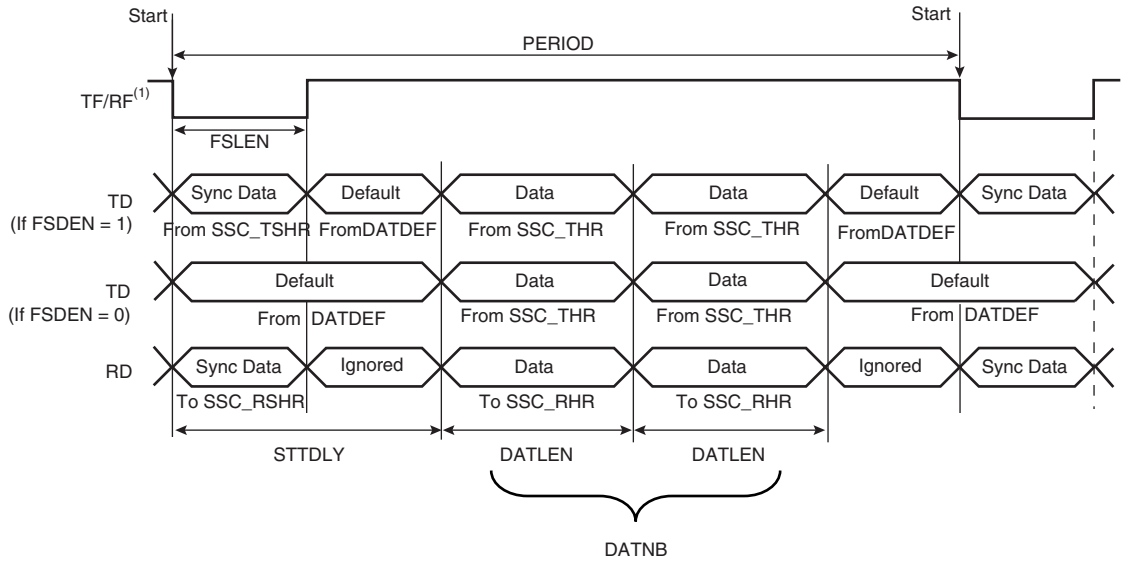
Table 72. 数据帧寄存器

发送器	接收器	域	长度	注释
SSC_TFMR	SSC_RFMR	DATLEN	达到 32	字长
SSC_TFMR	SSC_RFMR	DATNB	达到 16	发送器帧字数
SSC_TFMR	SSC_RFMR	MSBF		1 高位在先
SSC_TFMR	SSC_RFMR	FSLEN	达到 16	同步数据寄存器大小
SSC_TFMR		DATDEF	0 或 1	数据默认值结束

Table 72. 数据帧寄存器

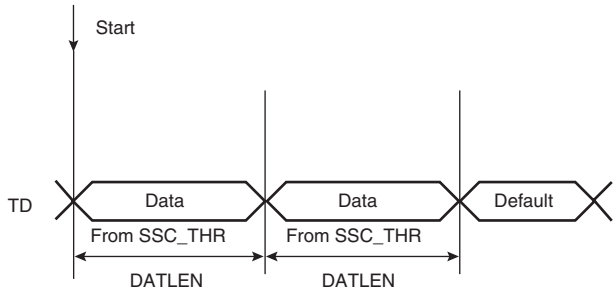
发送器	接收器	域	长度	注释
SSC_TFMR		FSDEN		使能发送 SSC_TSHR
SSC_TCMR	SSC_RCMR	PERIOD	达到 512	帧大小
SSC_TCMR	SSC_RCMR	STTDLY	达到 255	发送启动延迟大小

Figure 142. 边沿 / 脉冲启动模式下发送与接收帧格式



Note: 1. TF/RF 输入下降沿示例。

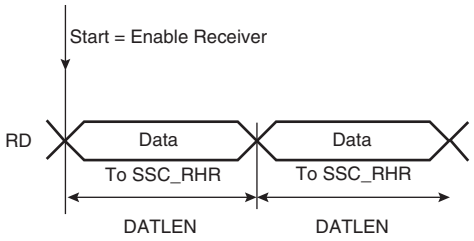
Figure 143. 连续模式下发送帧格式



Start: 1. TXEMPTY set to 1
2. Write into the SSC_THR

Note: 1. STTDLY 设置为 0。本例中，SSC_THR 载入两次。FSDEN 值对发送无效。连续模式下不能输出同步数据。

Figure 144. 连续模式下接收帧格式



Note: 1. STTDLY 设置为 0。

循环模式

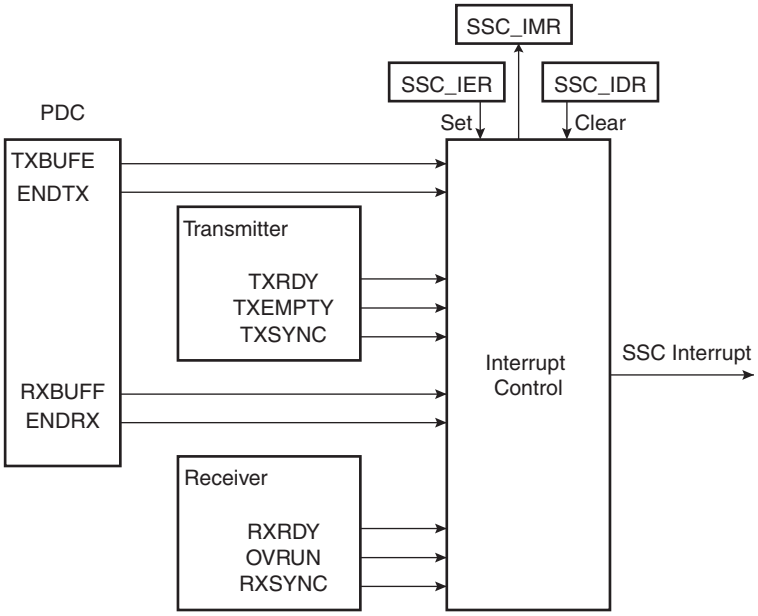
接收器可编程为接收发送器发送的数据。通过设定 SSC_RFMR 中循环模式 (LOOP) 位实现。此时，RD 与 TD 连接，RF 与 TF 连接，RK 与 TK 连接。

中断

SSC_SR 中大部分位在中断管理寄存器中有对应位。

SSC控制器可编程为当其检测到事件时产生中断。中断控制通过写SSC_IER (中断使能寄存器) 及 SSC_IDR (中断禁用寄存器) 来实现，二者通过在 SSC_IMR (中断屏蔽寄存器) 的相应位置位或清零来分别使能与禁用相应中断，SSC_IMR 通过在与 AIC 连接的 SSC 中断线上出现来产生中断。

Figure 145. 中断框图



SSC 应用示例

SSC 支持用于音频或高速串行链接的几种串行通信模式。下面给出一些标准应用。所有 SSC 支持的串行链接应用在此处未列出。

Figure 146. 音频应用框图

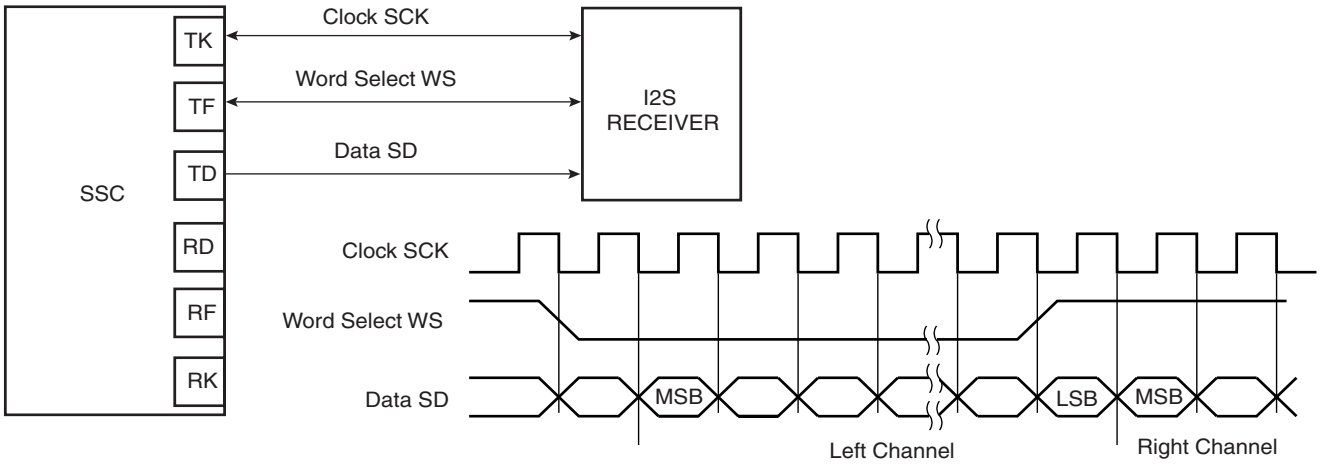


Figure 147. 编译码器应用框图

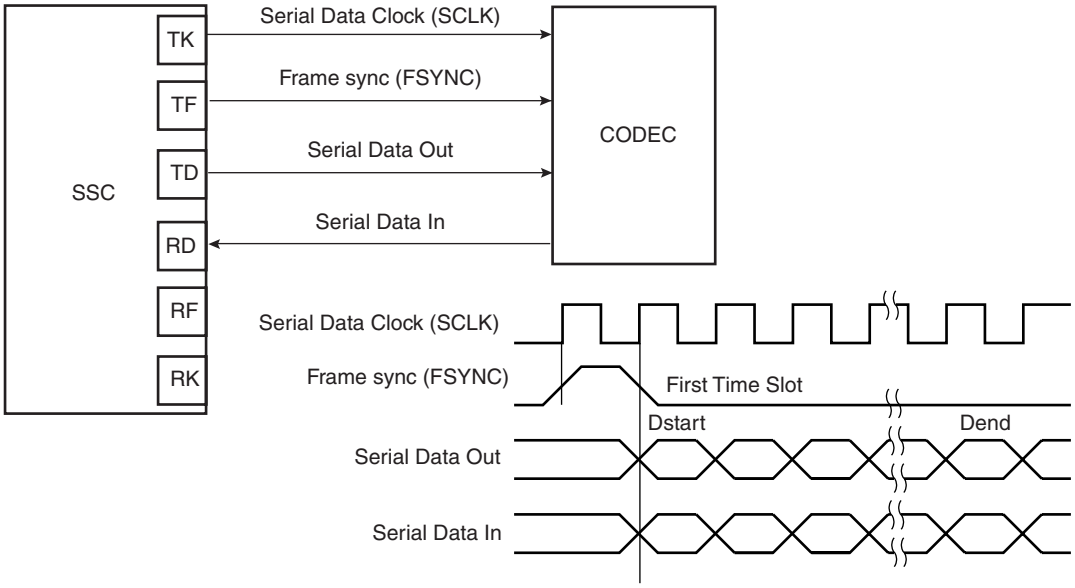
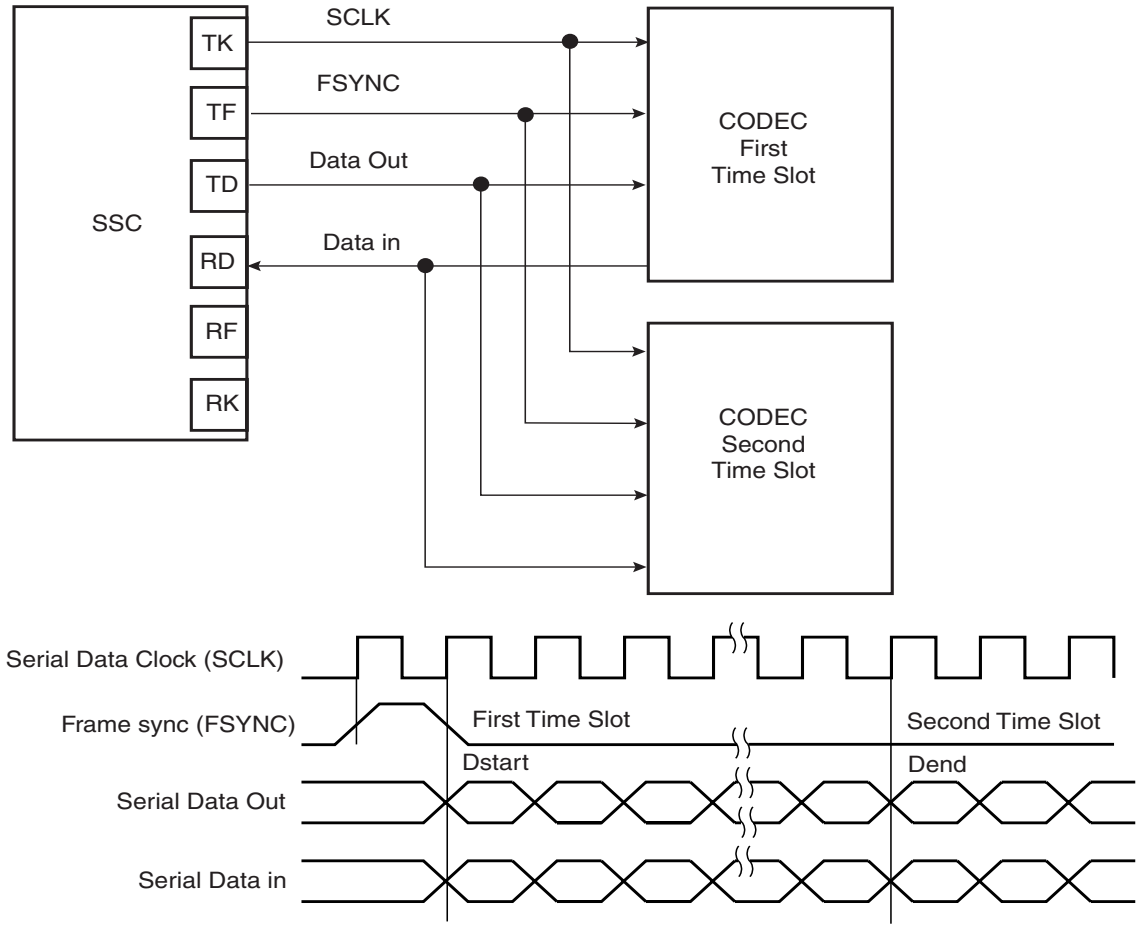


Figure 148. 时间插槽应用框图



同步串行控制器 (SSC) 用户接口

Table 73. 同步串行控制器 (SSC) 寄存器映射

偏移	寄存器	寄存器名称	访问类型	复位值
0x0	控制寄存器	SSC_CR	写	–
0x4	时钟模式寄存器	SSC_CMR	读 / 写	0x0
0x8	保留	–	–	–
0xC	保留	–	–	–
0x10	接收时钟模式寄存器	SSC_RCMR	读 / 写	0x0
0x14	接收帧模式寄存器	SSC_RFMR	读 / 写	0x0
0x18	发送时钟模式寄存器	SSC_TCMR	读 / 写	0x0
0x1C	发送帧模式寄存器	SSC_TFMR	读 / 写	0x0
0x20	接收保持寄存器	SSC_RHR	读	0x0
0x24	发送保持寄存器	SSC_THR	写	–
0x28	保留	–	–	–
0x2C	保留	–	–	–
0x30	接收同步保持寄存器	SSC_RSHR	读	0x0
0x34	发送同步保持寄存器	SSC_TSHR	读 / 写	0x0
0x38	保留	–	–	–
0x3C	保留	–	–	–
0x40	状态寄存器	SSC_SR	读	0x000000CC
0x44	中断使能寄存器	SSC_IER	写	–
0x48	中断禁用寄存器	SSC_IDR	写	–
0x4C	中断屏蔽寄存器	SSC_IMR	读	0x0
0x50-0xFC	保留	–	–	–
0x100 - 0x124	为外设数据控制器 (PDC) 保留	–	–	–

SSC 控制寄存器

寄存器名称： SSC_CR

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
SWRST	–	–	–	–	–	TXDIS	TXEN
7	6	5	4	3	2	1	0
–	–	–	–	–	–	RXDIS	RXEN

- **RXEN: 接收使能**

0：无效。

1：若 RXDIS 未置位，使能数据接收 (1)。

- **RXDIS: 接收禁用**

0：无效。

1：禁用数据接收 (1)。

- **TXEN: 发送使能**

0：无效。

1：若 TXDIS 未置位，使能数据发送 (1)。

- **TXDIS: 发送禁用**

0：无效。

1：禁用数据发送 (1)。

- **SWRST: 软件复位**

0：无效。

1：执行软件复位。比 SSC_CR 其它位优先。

Note: 1. 只有数据管理受影响。

SSC 时钟模式寄存器

寄存器名称： SSC_CMR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	DIV			
7	6	5	4	3	2	1	0
DIV							

- **DIV: 时钟分频器**

0：时钟分频器无效。

其它任意值：分频时钟等于主机时钟除以 2 倍的 DIV。最大位速率为 MCK/2 ；最小位速率为 MCK/2 x 4095 = MCK/8190。

SSC 接收时钟模式寄存器

寄存器名称： SSC_RCMR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
PERIOD							
23	22	21	20	19	18	17	16
STTDLY							
15	14	13	12	11	10	9	8
–	–	–	–	START			
7	6	5	4	3	2	1	0
–	–	CKI	CKO			CKS	

- **CKS: 接收时钟选择**

CKS	接收时钟选择
0x0	分频时钟
0x1	TK 时钟信号
0x2	RK 引脚
0x3	保留

- **CKO: 接收时钟输出模式选择**

CKO	接收时钟输出模式	RK 引脚
0x0	无	单输入
0x1	连续接收时钟	输出
0x2-0x7	保留	

- **CKI: 接收时钟反转**

0：数据及帧同步信号在接收时钟下降沿采样。

1：数据及帧同步信号在接收时钟下降沿移出。

CKI 不影响 RK 输出时钟信号。

- **START: 接收启动选择**

START	接收启动
0x0	连续，一旦接收器使能，在前一数据传输结束后立即开始
0x1	发送启动
0x2	检测 RF 输入低电平
0x3	检测 RF 输入高电平
0x4	检测 RF 输入下降沿
0x5	检测 RF 输入上升沿
0x6	检测 RF 输入电平变化
0x7	检测 RF 输入任意边沿
0x8-0xF	保留

- **STTDLY: 接收启动延迟**

若 STTDLY 不为 0，在启动事件与实际开始接收间插入一个 STTDLY 时钟周期的延迟。当接收器编程与发送器同步时，延迟依然有效。

请注意：设置 STTDLY 时需非常小心。若必须设置 STTDLY，应考虑与其相关的 TAG (接收同步数据) 接收。

- **PERIOD: 接收周期分频器选择**

该域选择应用于所选时钟的分频器，以产生一个新的帧同步信号。若为 0，不产生 PERIOD 信号；若不为 0，每 2 x (PERIOD+1) 个接收时钟产生一个 PERIOD 信号。

SSC 接收帧模式寄存器

寄存器名称： SSC_RFMR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	FSEDGE
23	22	21	20	19	18	17	16
–	FSOS			FSLEN			
15	14	13	12	11	10	9	8
–	–	–	–	DATNB			
7	6	5	4	3	2	1	0
MSBF	–	LOOP	DATLEN				

- **DATLEN: 数据长度**

不支持 0x0。 DATLEN 值可设定在 0x1 到 0x1F 间。

位串包含 DATLEN + 1 数据位。此外，它定义了接收器 PDC 执行的传输大小。

若 DATLEN 小于或等于 7，数据以字节传输；若 DATLEN 在 8 到 15 间 (包括 15)，数据以半字传输；其它值时，数据以 32 位字传输。

- **LOOP: 循环模式**

0：普通工作模式。

1：RD 由 TD 驱动， RF 由 TF 驱动， TK 驱动 RK。

- **MSBF: 高位在先**

0：首先对数据寄存器中的低位采样。

1：首先对数据寄存器中的高位采样。

- **DATNB: 每帧数据数**

该域定义每次发送开始后接收的数据字数。若为 0，只传输 1 个数据字，最多可传输 16 个数据字。

- **FSLEN: 接收帧同步长度**

该域定义接收帧同步信号长度及采样并保存于接收同步数据寄存器中的数据位数。只有当 FSOS 在正脉冲或负脉冲时设置才有效。

- **FSOS: 接收帧同步输出选择**

FSOS	选定的接收帧同步信号	RF 引脚
0x0	无	单输入
0x1	负脉冲	输出
0x2	正脉冲	输出
0x3	数据传输时拉低	输出
0x4	数据传输时拉高	输出
0x5	每次数据传输启动时切换	输出
0x6-0x7	保留	未定义

- **FSEDGE: 帧同步边沿检测**

确定帧同步边沿，在 SSC 状态寄存器的 RXSYN 中设置。

FSEDGE	帧同步边沿检测
0x0	正沿检测
0x1	负沿检测

SSC 发送时钟模式寄存器

寄存器名称： SSC_TCMR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
PERIOD							
23	22	21	20	19	18	17	16
STTDLY							
15	14	13	12	11	10	9	8
–	–	–	–	START			
7	6	5	4	3	2	1	0
–	–	CKI	CKO			CKS	

- **CKS: 发送时钟选择**

CKS	选定的发送时钟
0x0	分频时钟
0x1	RK 时钟信号
0x2	TK 引脚
0x3	保留

- **CKO: 发送时钟输出模式选择**

CKO	发送时钟输出模式	TK 引脚
0x0	无	单输入
0x1	连续发送时钟	输出
0x2-0x7	保留	

- **CKI: 发送时钟反转**

0：发送时钟下降沿数据与帧同步信号移出。

1：发送时钟上升沿数据与帧同步信号移出。

CKI 只影响发送时钟，对输出时钟信号无影响。

- **START: 发送启动选择**

START	发送启动
0x0	连续，一旦将字写入 SSC_THR 寄存器 (若发送使能) ，在前一个数据发送结束后立即启动
0x1	接收启动
0x2	检测 TF 低电平信号
0x3	检测 TF 高电平信号
0x4	检测 TF 下降沿信号
0x5	检测 TF 上升沿信号
0x6	检测 TF 电平变化信号
0x7	检测 TF 边沿信号
0x8-0xF	保留

- **STTDLY: 发送启动延迟**

若 STTDLY 不为 0，在启动事件与实际开始接收间插入一个 STTDLY 时钟周期的延迟。当发送器编程与接收器同步时，延迟依然有效。

请注意：设置 STTDLY 时需非常小心。若 STTDLY 与 TAG (发送同步数据) 相比太短，数据取代 TAG 结束发射。

- **PERIOD: 发送周期分频器选择**

该域选择应用于所选时钟的分频器，以产生一个新的帧同步信号。若为 0，不产生 PERIOD 信号；若不为 0，每 2 x (PERIOD+1) 个发送时钟产生一个 PERIOD 信号。

SSC 发送帧模式寄存器

寄存器名称： SSC_TFMR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	FSEDGE
23	22	21	20	19	18	17	16
FSDEN	FSOS			FSLEN			
15	14	13	12	11	10	9	8
–	–	–	–	DATNB			
7	6	5	4	3	2	1	0
MSBF	–	DATDEF	DATLEN				

- **DATLEN: 数据长度**

不支持 0x0。 DATLEN 值可设定在 0x1 到 0x1F 间。

位串包含 DATLEN + 1 数据位。此外，它定义了接收器 PDC 执行的传输大小。

若 DATLEN 小于或等于 7，数据以字节传输；若 DATLEN 在 8 到 15 间 (包括 15)，数据以半字传输；其它值时，数据以 32 位字传输。

- **DATDEF: 数据默认值**

该位定义发送输出时 TD 引脚电平。注意若引脚由 PIO 控制器定义为多驱动模式，该引脚只有在 SCC TD 输出为 1 时使能。

- **MSBF: 高位在先**

0：首先是数据寄存器中的低位移出。

1：首先是数据寄存器中的高位移出。

- **DATNB: 每帧数据数**

该域定义每次发送开始后发送的数据字数。若为 0，只传输 1 个数据字，最多可传输 16 个数据字。

- **FSLEN: 发送帧同步长度**

若 FSDEN 为 1 ,该域定义发送帧同步信号长度及发送同步数据寄存器中移出的数据位数；若为 0 ,在一个发送时钟周期中发送帧同步信号，并且最多可达 16 个时钟周期脉冲长度。

- **FSOS: 发送帧同步输出选择**

FSOS	选定的发送帧同步信号	TF 引脚
0x0	无	单输入
0x1	负脉冲	输出
0x2	正脉冲	输出
0x3	数据传输时拉低	输出
0x4	数据传输时拉高	输出
0x5	数据传输时切换	输出
0x6-0x7	保留	未定义

- **FSDEN: 帧同步数据使能**

0：发送帧同步信号时 TD 线驱动到默认值。

1：发送帧同步信号传输时移出 SSC_TSHR 值。

- **FSEDGE: 帧同步边沿检测**

确定帧同步边沿，在 SSC 状态寄存器的 TXSYN 中设置。

FSEDGE	帧同步边沿检测
0x0	正沿检测
0x1	负沿检测

SSC 接收保持寄存器

寄存器名称： SSC_RHR

访问类型： 只读

31	30	29	28	27	26	25	24
RDAT							
23	22	21	20	19	18	17	16
RDAT							
15	14	13	12	11	10	9	8
RDAT							
7	6	5	4	3	2	1	0
RDAT							

- **RDAT: 接收数据**

不管 SSC_RFMR 寄存器中 DATLEN 定义的数据位数为多少均右对齐。

SSC 发送保持寄存器

寄存器名称： SSC_THR

访问类型： 只写

31	30	29	28	27	26	25	24
TDAT							
23	22	21	20	19	18	17	16
TDAT							
15	14	13	12	11	10	9	8
TDAT							
7	6	5	4	3	2	1	0
TDAT							

TDAT: 发送数据

不管 SSC_TFMR 寄存器中 DATLEN 定义的数据位数为多少均右对齐。

SSC 接收同步保持寄存器

寄存器名称： SSC_RSHR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
RSDAT							
7	6	5	4	3	2	1	0
RSDAT							

- **RSDAT: 接收同步数据**

不管 SSC_RFMR 寄存器中 FSLEN 定义的数据位数为多少均右对齐。

SSC 发送同步保持寄存器

寄存器名称： SSC_TSHR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
TSDAT							
7	6	5	4	3	2	1	0
TSDAT							

- **TSDAT: 发送同步数据**

不管 SSC_TFMR 寄存器中 FSLEN 定义的数据位数为多少均右对齐。

SSC 状态寄存器

寄存器名称： SSC_SR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	RXEN	TXEN
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	–	–
7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: 发送就绪**

0：数据载入 SSC_THR 并等待载入发送移位寄存器。

1：SSC_THR 为空。

- **TXEMPTY: 发送空**

0：数据仍在 SSC_THR 中或正从发送移位寄存器发送。

1：写入 SSC_THR 的最后一个数据已载入发送移位寄存器并被其送出。

- **ENDTX: 发送结束**

0：上次写 SSC_TCR 或 SSC_TNCR 后寄存器 SSC_TCR 未达到 0。

1：上次写 SSC_TCR 或 SSC_TNCR 后寄存器 SSC_TCR 已达到 0。

- **TXBUFE: 发送缓冲器空**

0：SSC_TCR 或 SSC_TNCR 值不为 0。

1：SSC_TCR 与 SSC_TNCR 值均为 0。

- **RXRDY: 接收就绪**

0：SSC_RHR 为空。

1：数据已被接收并载入 SSC_RHR。

- **OVRUN: 接收溢出**

0：上次读状态寄存器后，前一数据未被读取时无数据载入 SSC_RHR。

1：上次读状态寄存器后，前一数据未被读取时已有数据载入 SSC_RHR。

- **ENDRX: 接收结束**

0：数据写入接收计数寄存器或接收下一计数寄存器。

1：当接收计数寄存器达到 0， PDC 传输结束。

- **RXBUFF: 接收缓冲器满**

0：SSC_RCR 或 SSC_RNCR 值不为 0。

1：SSC_RCR 与 SSC_RNCR 值均为 0。

- **TXSYN: 发送同步**

0：上次读状态寄存器后未出现 Tx 同步。

1：上次读状态寄存器后出现 Tx 同步。

- **RXSYN: 接收同步**

0：上次读状态寄存器后未出现 Rx 同步。

1：上次读状态寄存器后出现 Rx 同步。

- **TXEN: 发送使能**

0：发送数据禁用。

1：发送数据使能。

- **RXEN: 接收使能**

0：接收数据禁用。

1：接收数据使能。

SSC 中断使能寄存器

寄存器名称 :SSC_IER

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	–	–
7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: 发送就绪**
- **TXEMPTY: 发送空**
- **ENDTX: 发送结束**
- **TXBUFE: 发送缓冲器空**
- **RXRDY: 接收就绪**
- **OVRUN: 接收移出**
- **ENDRX: 接收结束**
- **RXBUFF: 接收缓冲器满**
- **TXSYN: Tx 同步**
- **RXSYN: Rx 同步**

0：无效。

1：使能相应中断。

SSC 中断禁用寄存器

寄存器名称： SSC_IDR

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	–	–
7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: 发送就绪**
- **TXEMPTY: 发送空**
- **ENDTX: 发送结束**
- **TXBUFE: 发送缓冲器空**
- **RXRDY: 接收就绪**
- **OVRUN: 接收移出**
- **ENDRX: 接收结束**
- **RXBUFF: 接收缓冲器满**
- **TXSYN: Tx 同步**
- **RXSYN: Rx 同步**

0：无效。

1：禁用相应中断。

SSC 中断屏蔽寄存器

寄存器名称： SSC_IMR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	RXSYN	TXSYN	–	–
7	6	5	4	3	2	1	0
RXBUFF	ENDRX	OVRUN	RXRDY	TXBUFE	ENDTX	TXEMPTY	TXRDY

- **TXRDY: 发送就绪**
- **TXEMPTY: 发送空**
- **ENDTX: 发送结束**
- **TXBUFE: 发送缓冲器空**
- **RXRDY: 接收就绪**
- **OVRUN: 接收移出**
- **ENDRX: 接收结束**
- **RXBUFF: 接收缓冲器满**
- **TXSYN: Tx 同步**
- **RXSYN: Rx 同步**

0：相应中断禁用。

1：相应中断使能。

定时器 / 计数器 (TC)

概述

定时器 / 计数器 (TC) 包括三个 相同的 16 位定时器 / 计数器通道。

每个通道可独立编程，以完成不同功能，包括 : 频率测量、事件计数、间隔测量、脉冲产生、延迟时间及脉宽调制。

每个通道有三个外部时钟输入，五个内部时钟输入及两个可由用户配置的多功能输入 / 输出信号。每个通道驱动一个可编程内部中断信号来产生处理器中断。

定时器 / 计数器有两个作用于这三个 TC 通道的全局寄存器。

块控制寄存器允许使用同样的指令同时启动三个通道。

块模式寄存器为每个通道定义外部时钟输入，允许将它们链接。

方框图

Figure 149. 定时器 / 计数器框图

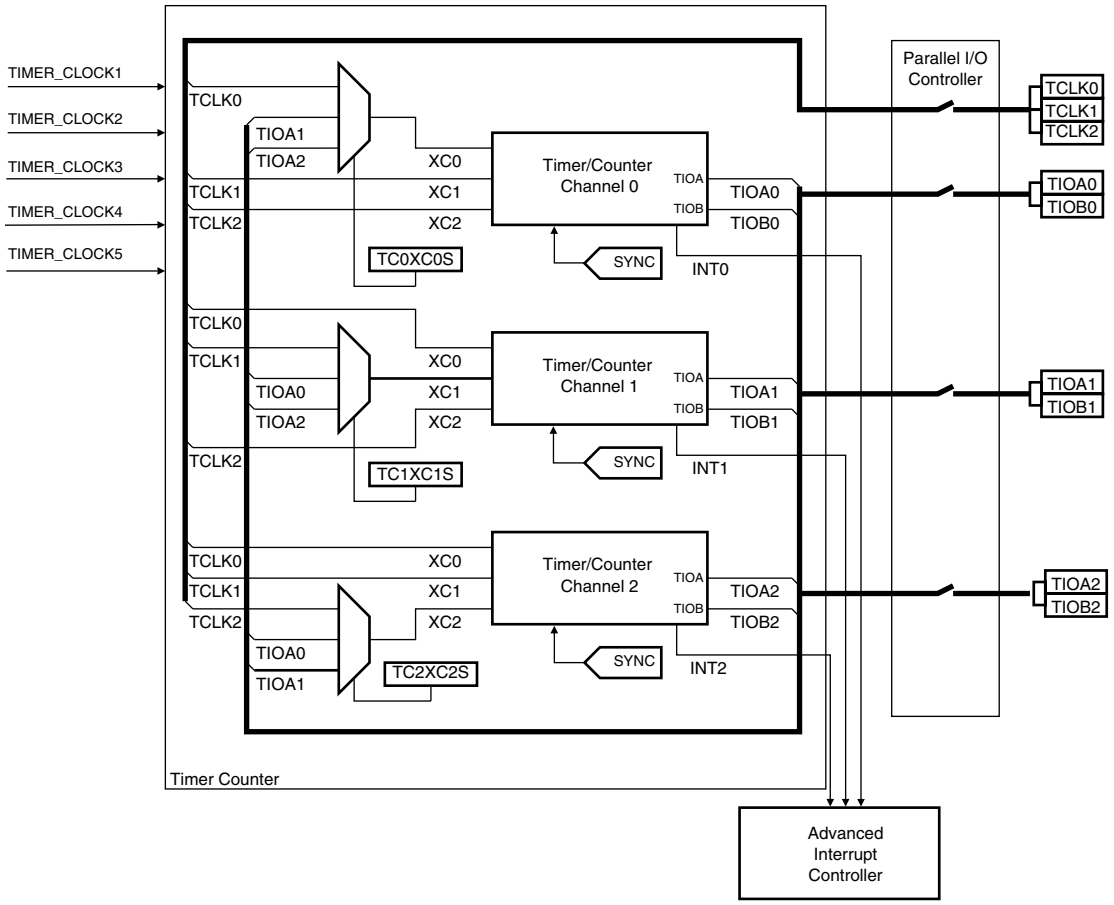


Table 74. 信号名称说明

块 / 通道	信号名称	说明
通道信号	XC0, XC1, XC2	外部时钟输入
	TIOA	捕获模式：定时器 / 计数器输入 波形模式：定时器 / 计数器输出
	TIOB	捕获模式：定时器 / 计数器输入 波形模式：定时器 / 计数器输入 / 输出
	INT	中断信号输出
	SYNC	同步输入信号

引脚名称列表

Table 75. TC 引脚列表

引脚名称	说明	类型
TCLK0-TCLK2	外部时钟输入	输入
TIOA0-TIOA2	I/O 线 A	I/O
TIOB0-TIOB2	I/O 线 B	I/O

附属产品

I/O 线

连接外设的引脚可与 PIO 线复用。必须先对 PIO 控制器编程，给 TC 引脚分配外设功能。

电源管理

TC 由电源管理控制器 (PMC) 定时，因此必须先配置 PMC 以使能定时器 / 计数器时钟。

中断

TC 有一条与高级中断控制器 (AIC) 连接的中断线。处理中断前需要先对 AIC 编程，然后再配置 TC。

功能说明

TC 说明

TC 的三个通道相互独立，但操作相同。通道寄存器编程见 Table 77 on page 360。

16 位计数器

每通道有一个 16 位寄存器。寄存器值在所选时钟每个正沿处自减。当计数器达到 0xFFFF 并转为 0x0000 时，表明发生溢出， TC_SR (状态寄存器) 中 COVFS 位置位。

计数器当前值可通过计数器值寄存器 TC_CV 实时读取。计数器由触发器复位。此时，计数器值在下一个选定时钟有效边沿时转为 0x0000。

时钟选择

在块级上，每个通道输入时钟通过对 TC_BMR (块模式) 编程可与外部输入 TCLK0、 TCLK1、 TCLK2 或可配置的 I/O 信号 TIOA0、 TIOA1、 TIOA2 连接，见 Figure 150。

每个通道可独立选择内部或外部计数器时钟源：

- 内部时钟源：TIMER_CLOCK1、 TIMER_CLOCK2、 TIMER_CLOCK3、 TIMER_CLOCK4、 TIMER_CLOCK5
- 外部时钟信号：XC0、 XC1 或 XC2

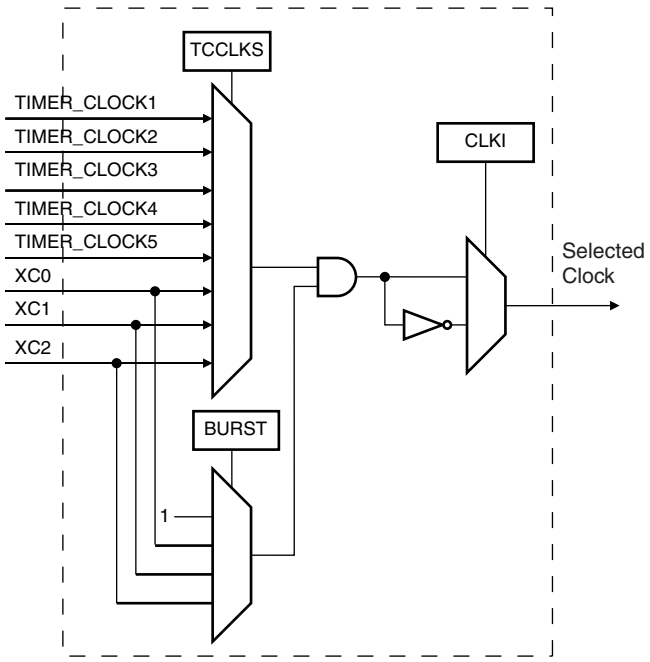
选择由 TC 通道模式寄存器的 TCCLKS 位完成。

所选时钟可通过 TC_CMR 寄存器的 CLKI 位实现反转。因此可使用时钟负沿进行计数。

脉冲功能使外部信号为高时时钟有效。模式寄存器中的 BURST 参数定义该信号（无、 XC0、 XC1、 XC2)。

Note: 使用外部时钟时，每个电平持续时间必须比主机时钟周期长。主机时钟频率至少为外部时钟频率的 2.5 倍。

Figure 150. 时钟选择

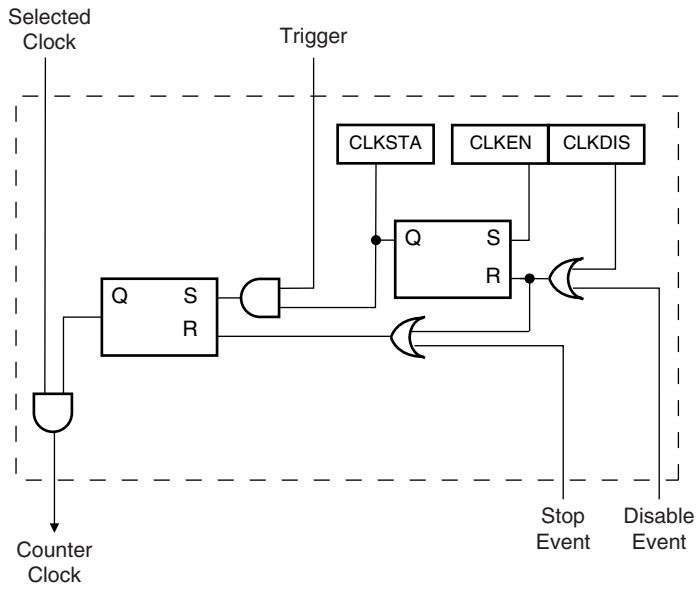


时钟控制

每个计数器时钟有两种控制方式：可使能 / 禁用或启动 / 停止，见 Figure 151。

- 用户可使用控制寄存器的 CLKEN 与 CLKDIS 命令使能或禁用时钟。捕获模式下，若 TC_CMR 中 LDBDIS 位置为 1 ,通过 RB 载入事件将时钟禁用。波形模式下 ,若 TC_CMR 中 CPCDIS 位置为 1，通过 RC 比较事件将时钟禁用。当时钟禁用时，启动与停止命令无效，只有控制寄存器的 CLKEN 命令可将时钟重新使能。当时钟使能后，置位状态寄存器的 CLKSTA 位。
- 时钟也可启动或停止：触发器 (软件、同步、外部或比较) 用来启动时钟。捕获模式下，时钟由 RB 载入事件停止 (TC_CMR 中 LDBSTOP = 1) ；波形模式下，时钟由 RC 比较事件停止 (TC_CMR 中 CPCSTOP = 1)。只有当时钟使能时启动与停止命令才有效。

Figure 151. 时钟控制



TC 操作模式

每个通道可工作在两种不同模式下：

- 捕获模式提供信号测量。
- 波形模式用来产生波形。

TC 操作模式由 TC 通道模式寄存器的 WAVE 位编程设定。

捕获模式下， TIOA 与 TIOB 配置为输入。

波形模式下， TIOA 配置为输出，若未选择外部触发器 TIOB 也为输出。

触发器

触发器复位计数器并启动计数器时钟。两种模式下有三种通用触发器，第四种外部触发器分别适用于每种模式。

下列触发器适用于两种模式：

- 软件触发器：每个通道有一软件触发器，通过设置 TC_CCR 中的 SWTRG 有效。
- SYNC：每个通道有一个同步信号 SYNC。当出现时，该信号与软件触发器效果相同。通过写 TC_BCR(块控制)，所有通道的 SYNC 信号同时出现。
- 比较 RC 触发器:RC 在每个通道中执行，若 TC_CMR 中 CPCTRG 置位，能在计数器值等于 RC 值时提供触发。

通道也能配置为有一个外部触发器。捕获模式下，在 TIOA 与 TIOB 信号间选择外部触发信号；波形模式下，外部事件可在下列信号上编程：TIOB、 XC0、 XC1 或 XC2。通过设置 TC_CMR 中的 ENETRG 可实现外部事件执行触发。

若使用外部触发器，脉冲持续时间必须比主机时钟周期长，以便对其检测。

不管触发器是否使用，它将在后续选定时钟的有效沿记录。即触发后计数器值即不为零，特别是当选定低频信号作为时钟时。

捕获工作模式

清除 TC_CMR (通道模式寄存器) 的 WAVE 参数即可进入该模式。

捕获模式允许TC通道对脉冲时间、频率、周期占空比及作为输入的TIOA与TIOB信号进行测量。

Figure 152 给出捕获模式下 TC 通道配置。

捕获寄存器 A 与 B

寄存器 A 与 B (RA 与 RB) 作为捕获寄存器使用。即当可编程事件在 TIOA 上出现时，它们将载入计数器值。

TC_CMR中的LDRA参数定义寄存器A的载入TIOA边沿；LDRB参数定义寄存器B的载入TIOA边沿。

只有在最近触发后未载入或 RB 已载入时， RA 才会载入。

RB 只有在 RA 已载入或最后载入 RB 时，才会载入。

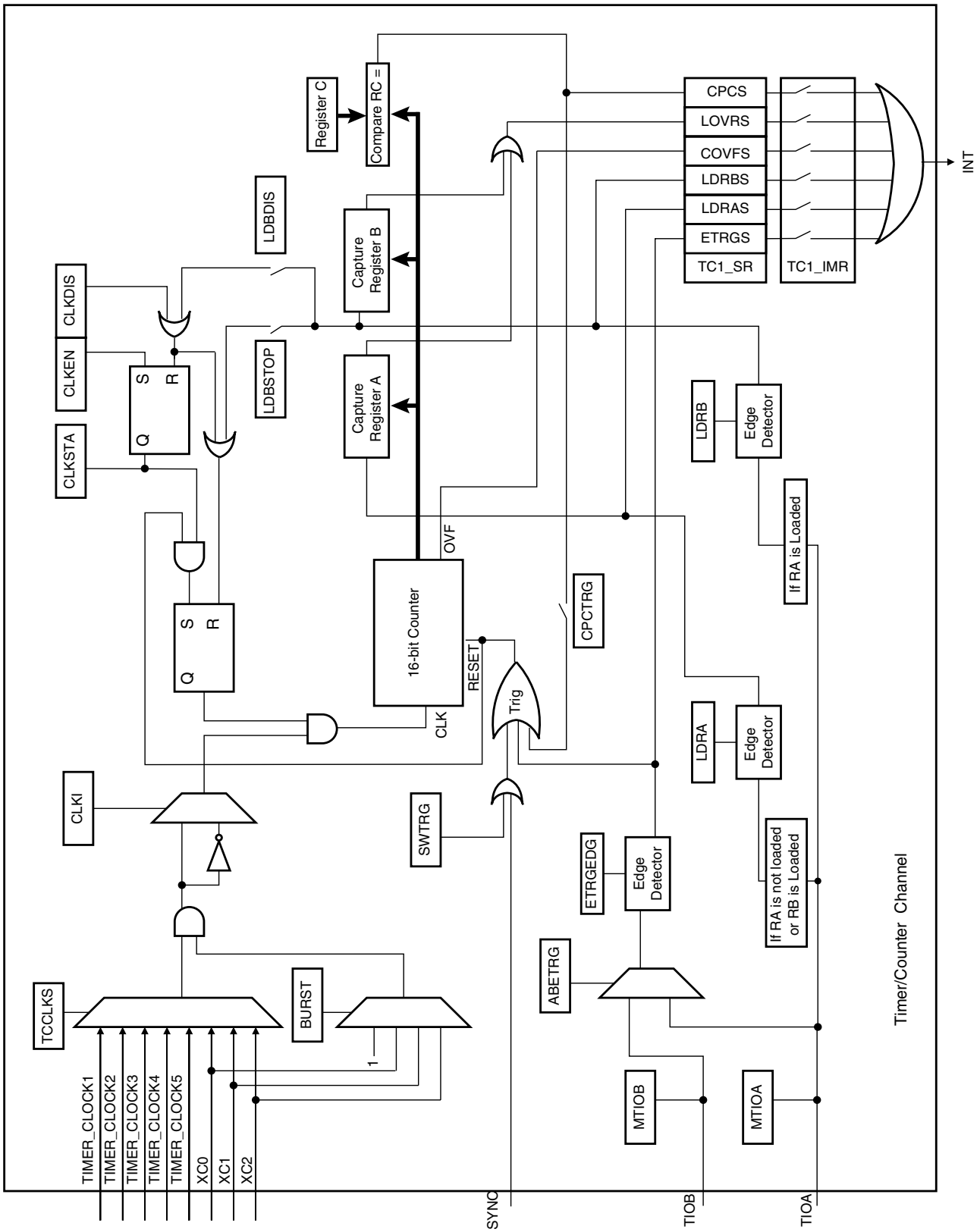
读最近载入值前载入 RA 或 RB 将设置 TC_SR (状态寄存器) 的溢出错误标志 (LOVRS)。此时将旧值覆盖。

触发条件

除 SYNC 信号、软件触发器及 RC 比较寄存器外，还可定义外部触发。

TC_CMR寄存器的ABETRG位选择使用TIOA或TIOB输入信号作为外部触发。ETRGEDG参数定义产生外部触发的检测边沿(上升沿、下降沿或两者皆可)。若ETRGEDG = 0 ,外部触发禁用。

Figure 152. 捕获模式



波形工作模式

通过设置 TC_CMR 寄存器的 WAVE 参数进入波形工作模式。

波形工作模式下，TC 通道产生 1 个或 2 个相同频率的可独立编程占空比的 PWM 信号，或产生不同类型的单发射或重复脉冲。

该模式下，TIOA 配置为输出，TIOB 在未使用外部事件时也定义为输出 (TC_CMR 中的 EEVT 参数)。

Figure 153 给出波形工作模式下的 TC 通道配置。

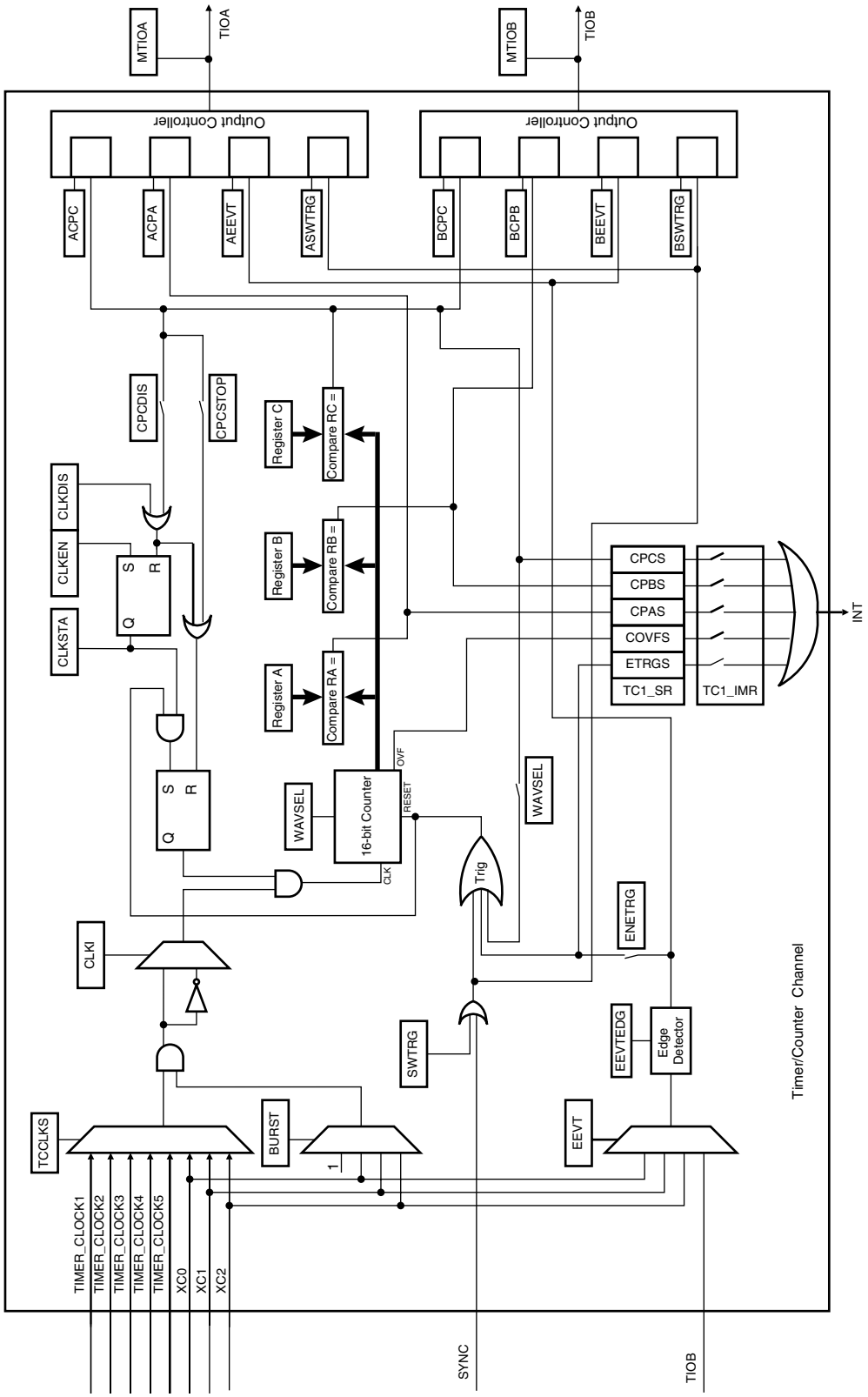
波形选择

根据 TC_CMR 中 WAVSEL 参数的不同，TC_CV 动作不同。

任何情况下，RA、 RB 及 RC 可作为比较寄存器使用。

RA比较器用来控制TIOA输出 ,RB比较用来控制TIOB输出(若配置正确) ,RC比较用来控制TIOA与 / 或 TIOB 输出。

Figure 153. 波形模式



WAVSEL = 00

当 WAVSEL = 00 时，TC_CV 值由 0 增加到 0xFFFF。一旦达到 0xFFFF，TC_CV 值复位。TC_CV 值重新增加且继续循环，见 Figure 154。

外部事件触发或软件触发可复位 TC_CV 值。注意，触发可能随时出现，见 Figure 155。

该配置下 RC 比较不能编程产生触发。同时 RC 比较可停止计数器时钟 (TC_CMR 中 CPCSTOP = 1) 与 / 或禁用计数器时钟 (TC_CMR 中 CPCDIS = 1)。

Figure 154. WAVSEL= 00 无触发

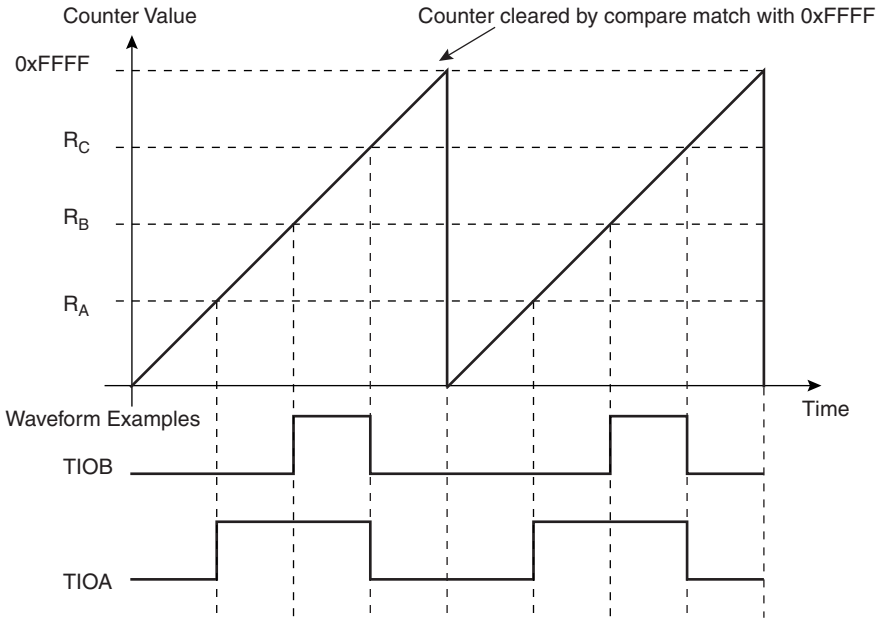
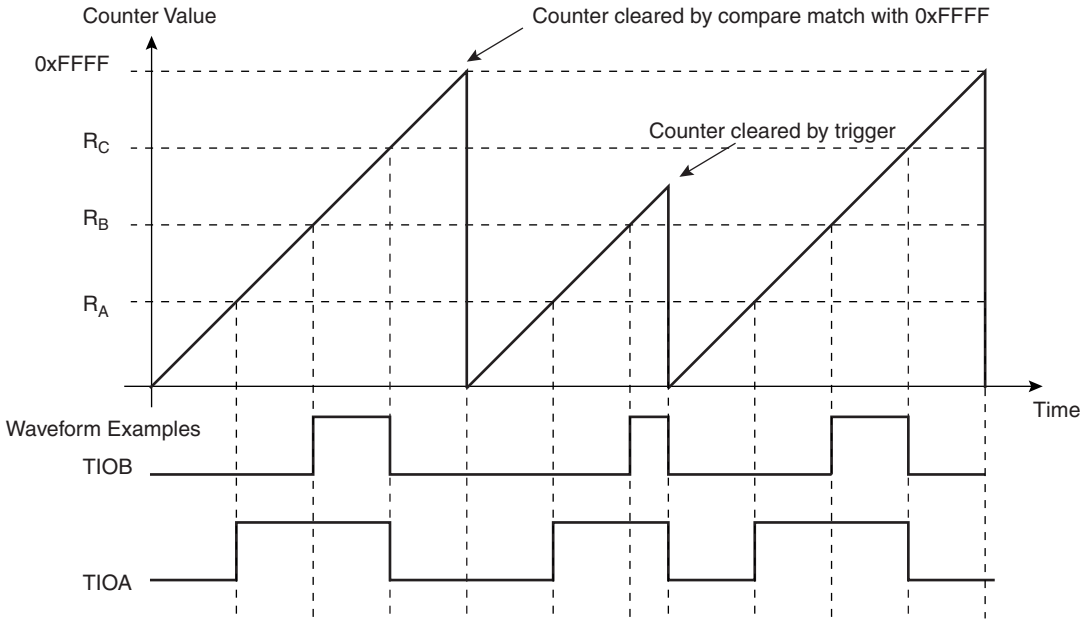


Figure 155. WAVSEL= 00 有触发



WAVSEL = 10

当 WAVSEL = 10 时，TC_CV 值由 0 增加到 RC 值，然后自动复位。一旦 TC_CV 值复位，它开始重新循环，见 Figure 156。

若外部事件或软件触发器编程正确， TC_CV 可随时复位，见 Figure 157。

此外 ,RC 比较可停止计数器时钟 (TC_CMR 中 CPCSTOP = 1) 与 / 或禁用计数器时钟 (TC_CMR 中 CPCDIS = 1)。

Figure 156. WAVSEL = 10 无触发

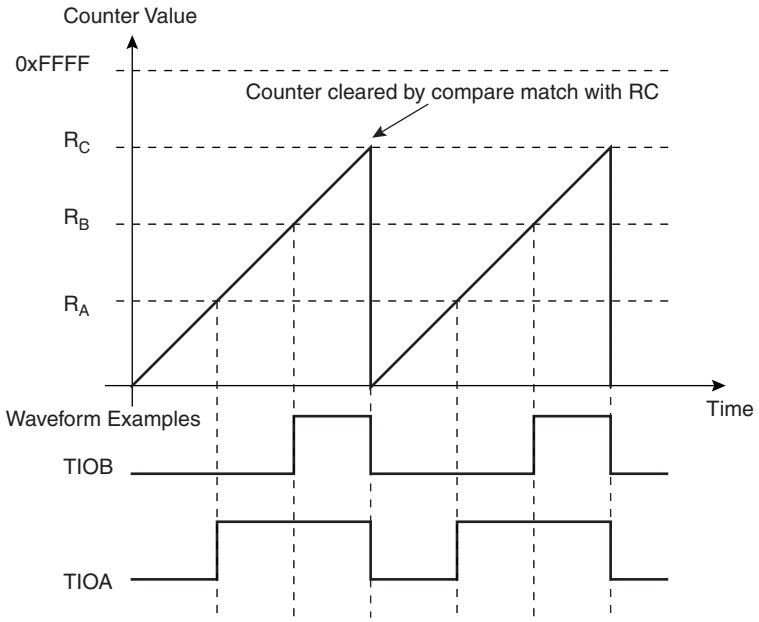
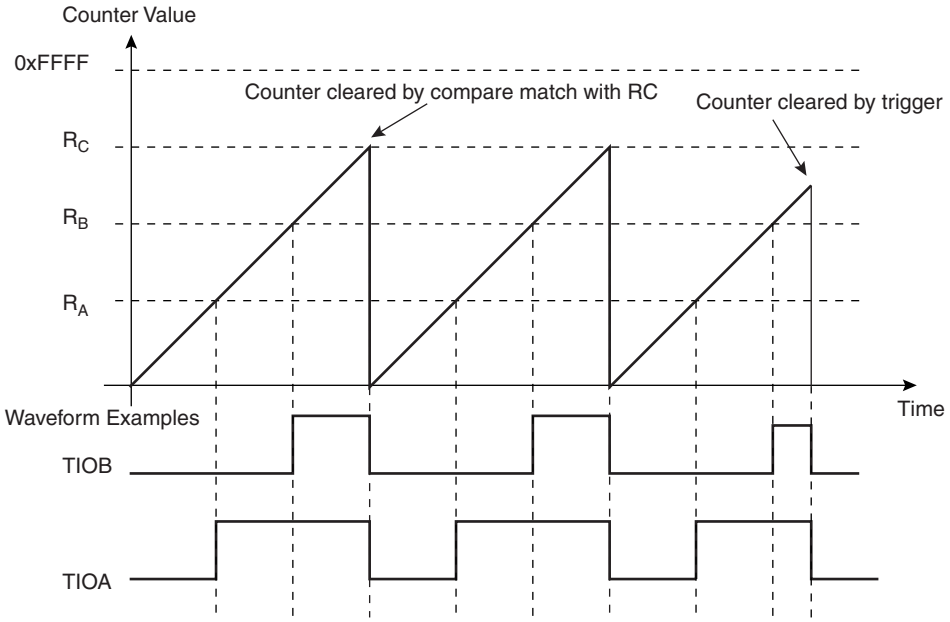


Figure 157. WAVSEL = 10 有触发



WAVSEL = 01

当 WAVSEL = 01， TC_CV 值由 0 增加到 0xFFFF。一旦达到 0xFFFF， TC_CV 值减到 0，再重新增加到 0xFFFF，如此循环下去，见 Figure 158。

外部事件或软件触发可随时修改 TC_CV。若 TC_CV 增加时触发出现， TC_CV 开始递减；若 TC_CV 递减时收到触发， TC_CV 开始递增，见 Figure 159。

该配置下，不可对 RC 比较编程产生触发。

RC 比较可停止计数器时钟 (CPCSTOP = 1) 与 / 或禁用计数器时钟 (CPCDIS = 1)。

Figure 158. WAVSEL = 01 无触发

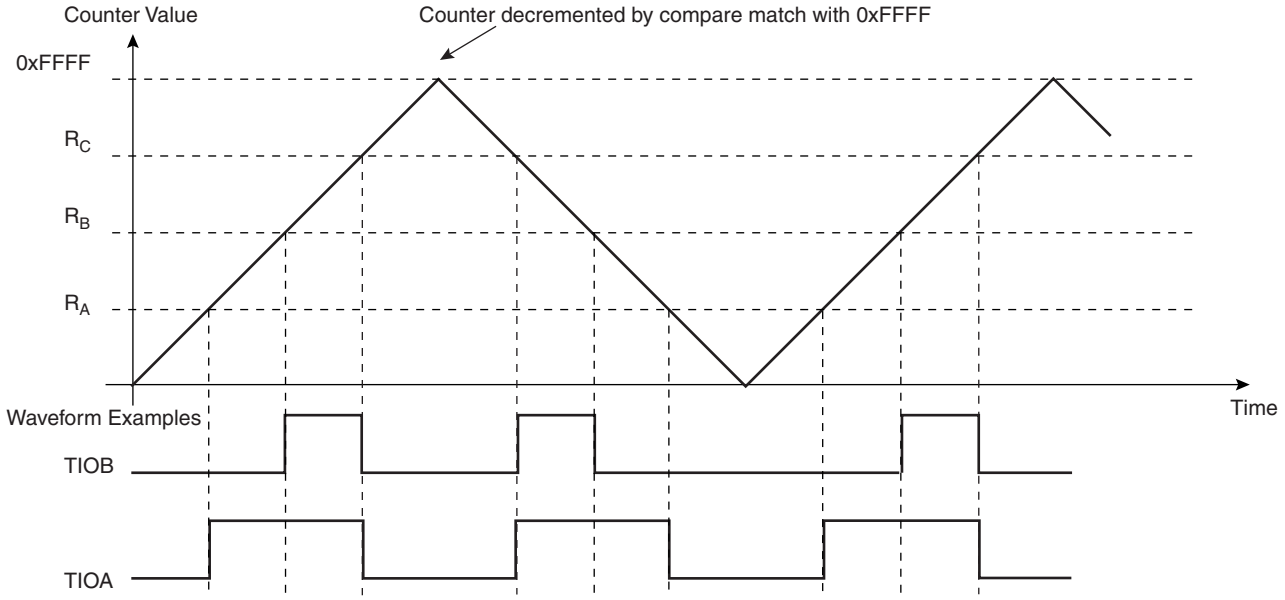
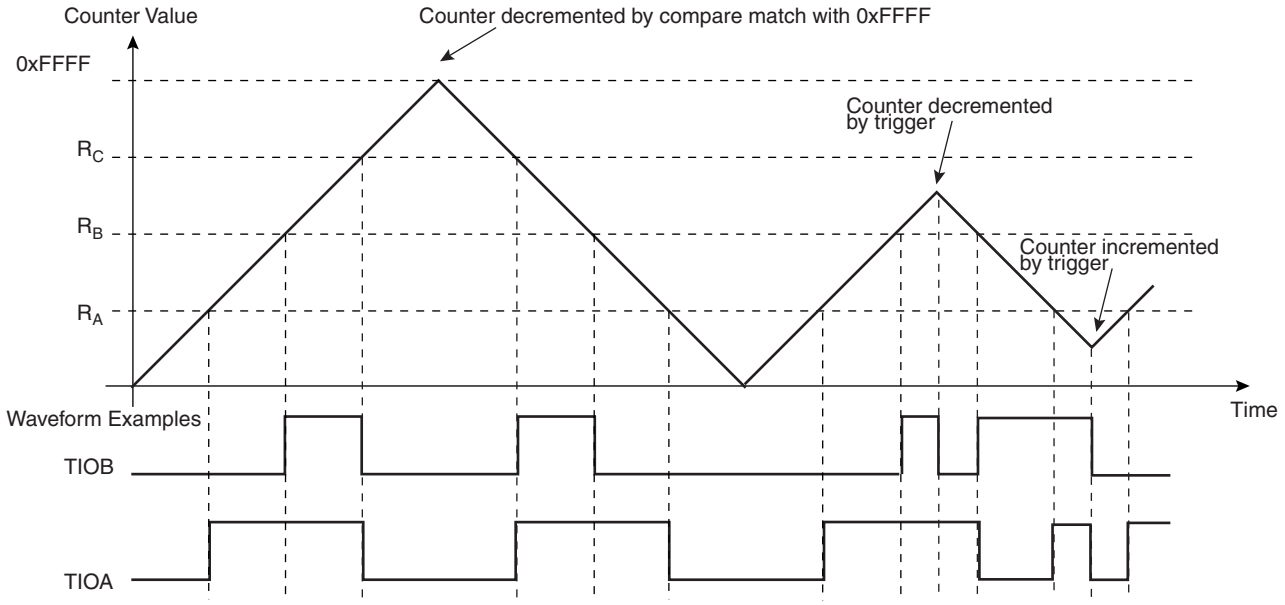


Figure 159. WAVSEL = 01 有触发



WAVSEL = 11

当 WAVSEL = 11， TC_CV 值由 0 增加到 RC。一旦达到 RC， TC_CV 值递减到 0，然后重新递增到 RC，如此循环下去，见 Figure 160。

外部事件或软件触发可随时修改 TC_CV。若 TC_CV 增加时触发出现， TC_CV 开始递减；若 TC_CV 递减时收到触发， TC_CV 开始递增，见 Figure 161。

RC 比较可停止计数器时钟 (CPCSTOP = 1) 与 / 或禁用计数器时钟 (CPCDIS = 1)。

Figure 160. WAVSEL = 11 无触发

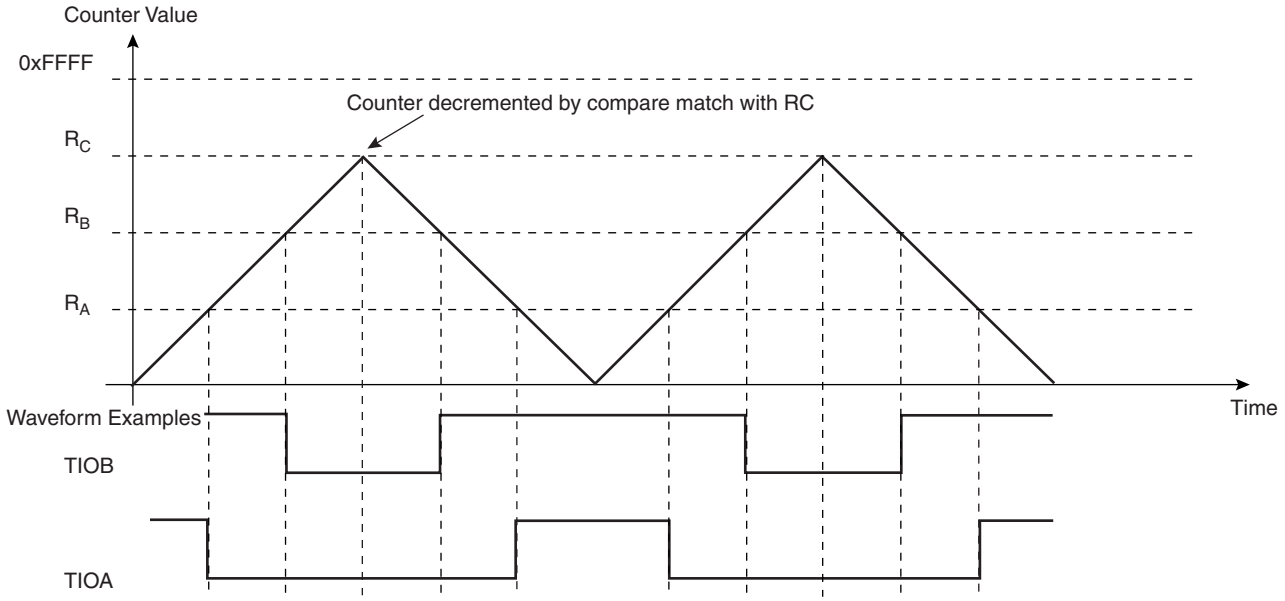
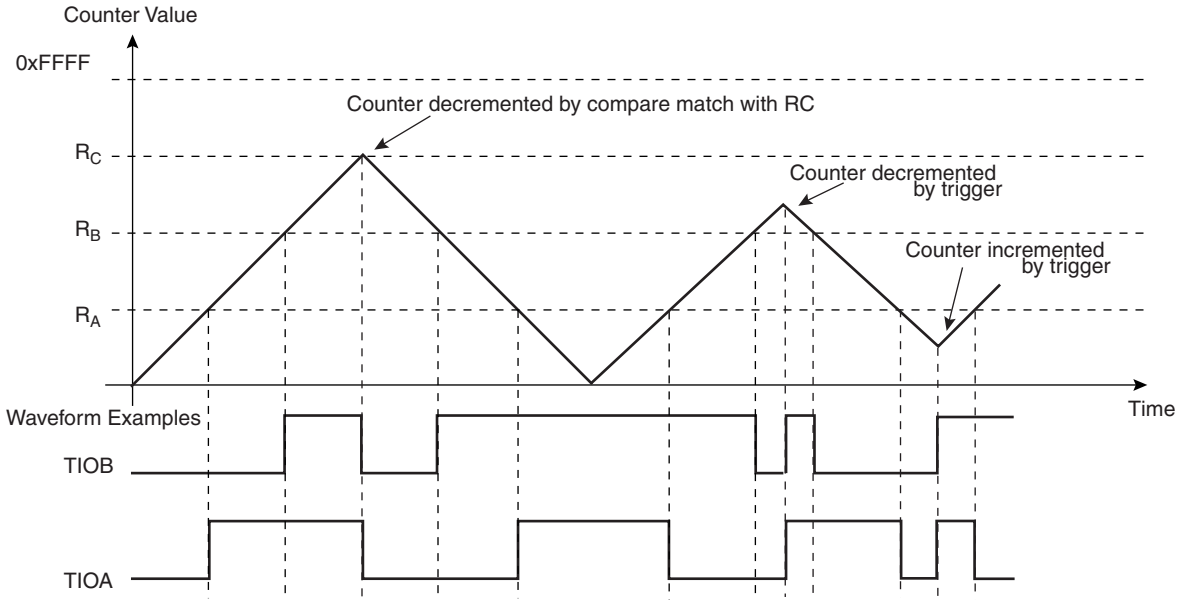


Figure 161. WAVSEL = 11 有触发



外部事件 / 触发条件

可对外部事件编程来检测时钟源 (XC0、XC1、XC2) 或 TIOB。然后可选择外部事件作为触发。

TC_CMR 中 EEVT 参数用来选择外部触发器。参数 EEVTEDG 定义每个可能的外部触发边沿 (上升沿、下降沿或二者皆可)。若 EEVTEDG 清零，不定义外部事件。

若定义 TIOB 为外部事件信号 (EEVT = 0) ,TIOB 不再作为输出且 TC 通道只可产生 TIOA 波形。

若定义了外部事件，通过设置 TC_CMR 中的 ENETRG 位，外部事件可作为触发器。

与捕获模式相同， SYNC 信号与软件触发作为触发器同样有效。若 WAVSEL 参数设置恰当，RC 比较也可作为触发器。

输出控制器

输出控制器定义事件后 TIOA 与 TIOB 输出电平变化。只有当 TIOB 定义为输出时使用 TIOB 控制 (不是外部事件)。

下列事件控制 TIOA 与 TIOB：软件触发器、外部事件及 RC 比较。RA 比较控制 TIOA，RB 比较控制 TIOB。根据相应的 TC_CMR 定义，每个事件可编程设置、清除或切换输出。

定时器 / 计数器 (TC) 用户接口

全局寄存器映射

Table 76. 定时器 / 计数器 (TC) 全局寄存器映射

偏移	通道 / 寄存器	名称	访问类型	复位值
0x00	TC 通道 0	见 Table 77		
0x40	TC 通道 1	见 Table 77		
0x80	TC 通道 2	见 Table 77		
0xC0	TC 块控制寄存器	TC_BCR	只写	–
0xC4	TC 块模式寄存器	TC_BMR	读 / 写	0

TC_BCR (块控制寄存器)与TC_BMR (块控制寄存器)控制整个TC块。TC通道由Table 77列出的寄存器控制。 Table 77 中每个通道寄存器偏移与 Table 77 中提到的相应通道偏移相关。

通道存储器映射

Table 77. 定时器 / 计数器 (TC) 通道存储器映射

偏移	寄存器	名称	访问类型	复位值
0x00	通道控制寄存器	TC_CCR	只写	–
0x04	通道模式寄存器	TC_CMR	读 / 写	0
0x08	保留	–	–	–
0x0C	保留	–	–	–
0x10	计数器值	TC_CV	只读	0
0x14	寄存器 A	TC_RA	读 / 写 (1)	0
0x18	寄存器 B	TC_RB	读 / 写 (1)	0
0x1C	寄存器 C	TC_RC	读 / 写	0
0x20	状态寄存器	TC_SR	只读	0
0x24	中断使能寄存器	TC_IER	只写	–
0x28	中断禁用寄存器	TC_IDR	只写	–
0x2C	中断屏蔽寄存器	TC_IMR	只读	0
0x30-0xFC	保留	–	–	–

Note: 1. WAVE = 0 时读。

TC 块控制寄存器

寄存器名称 : TC_BCR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	SYNC

- **SYNC: 同步命令**

0 = 无效。

1 = 出现 SYNC 信号，给每个通道同时产生软件触发。

TC 块模式寄存器

寄存器名称 : TC_BMR

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	TC2XC2S		TCXC1S		TC0XC0S	

- **TC0XC0S: 外部时钟信号 0 选择**

TC0XC0S		信号与 XC0 连接
0	0	TCLK0
0	1	无
1	0	TIOA1
1	1	TIOA2

- **TC1XC1S: 外部时钟信号 1 选择**

TC1XC1S		信号与 XC1 连接
0	0	TCLK1
0	1	无
1	0	TIOA0
1	1	TIOA2

- **TC2XC2S: 外部时钟信号 2 选择**

TC2XC2S		信号与 XC2 连接
0	0	TCLK2
0	1	无
1	0	TIOA0
1	1	TIOA1

TC 通道控制寄存器

寄存器名称 : TC_CCR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	SWTRG	CLKDIS	CLKEN

- **CLKEN: 计数器时钟使能命令**

0 = 无效。

1 = 若 CLKDIS 不为 1，使能时钟。

- **CLKDIS: 计数器时钟禁用命令**

0 = 无效。

1 = 禁用时钟。

- **SWTRG: 软件触发命令**

0 = 无效。

1 = 软件触发执行：计数器复位，时钟启动。

TC 通道模式寄存器：捕获模式

寄存器名称： TC_CMR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	LDRB		LDRA	
15	14	13	12	11	10	9	8
WAVE = 0	CPCTRG	–	–	–	ABETRG	ETRGEDG	
7	6	5	4	3	2	1	0
LDBDIS	LDBSTOP	BURST		CLKI	TCCLKS		

- **TCCLKS: 时钟选择**

TCCLKS			选定时钟
0	0	0	TIMER_CLOCK1
0	0	1	TIMER_CLOCK2
0	1	0	TIMER_CLOCK3
0	1	1	TIMER_CLOCK4
1	0	0	TIMER_CLOCK5
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

- **CLKI: 时钟反转**

0 = 时钟上升沿计数器增加。

1 = 时钟下降沿计数器增加。

- **BURST: 脉冲信号选择**

BURST		
0	0	时钟不是由外部时钟开启
0	1	XC0 与选定时钟相与
1	0	XC1 与选定时钟相与
1	1	XC2 与选定时钟相与

- **LDBSTOP: RB 载入时，计数器时钟停止**

0 = 当 RB 载入出现时，计数器时钟未停止。

1 = 当 RB 载入出现时，计数器时钟停止。

- **LDBDIS: RB 载入时，计数器时钟禁用**

0 = 当 RB 载入出现时，计数器时钟未禁用。

1 = 当 RB 载入出现时，计数器时钟禁用。

- **ETRGEDG: 外部触发边沿选择**

ETRGEDG		边沿
0	0	无
0	1	上升沿
1	0	下降沿
1	1	任意沿

- **ABETRG: TIOA 或 TIOB 外部触发选择**

0 = TIOB 用作外部触发器。

1 = TIOA 用作外部触发器。

- **CPCTRG: RC 比较触发使能**

0 = RC 比较对计数器及其时钟无效。

1 = RC 比较复位计数器并启动计数器时钟。

- **WAVE**

0 = 捕获模式使能。

1 = 捕获模式禁用 (波形模式使能)。

- **LDRA: RA 载入选择**

LDRA		边沿
0	0	无
0	1	TIOA 上升沿
1	0	TIOA 下降沿
1	1	TIOA 任意沿

- **LDRB: RB 载入选择**

LDRB		边沿
0	0	无
0	1	TIOA 上升沿
1	0	TIOA 下降沿
1	1	TIOA 任意沿

TC 通道模式寄存器：波形模式

寄存器名称： TC_CMR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
BSWTRG		BEEVT		BCPC		BCPB	
23	22	21	20	19	18	17	16
ASWTRG		AEEVT		ACPC		ACPA	
15	14	13	12	11	10	9	8
WAVE = 1	WAVSEL		ENETRG	EEVT		EEVTEDG	
7	6	5	4	3	2	1	0
CPCDIS	CPCSTOP	BURST		CLKI	TCCLKS		

- **TCCLKS: 时钟选择**

TCCLKS			选定时钟
0	0	0	TIMER_CLOCK1
0	0	1	TIMER_CLOCK2
0	1	0	TIMER_CLOCK3
0	1	1	TIMER_CLOCK4
1	0	0	TIMER_CLOCK5
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

- **CLKI: 时钟反转**

0 = 时钟上升沿计数器增加。

1 = 时钟下降沿计数器增加。

- **BURST: 脉冲信号选择**

BURST		
0	0	时钟不是由外部时钟开启
0	1	XC0 与选定时钟相与
1	0	XC1 与选定时钟相与
1	1	XC2 与选定时钟相与

- **CPCSTOP: RC 比较时，计数器时钟停止**

0 = 当计数器值达到 RC 时计数器时钟不停止。

1 = 当计数器值达到 RC 时计数器时钟停止。

- **CPCDIS: RC 比较时，计数器时钟禁用**

0 = 当计数器值达到 RC 时计数器时钟不禁用。

1 = 当计数器值达到 RC 时计数器时钟禁用。

- **EEVTEDG: 外部事件边沿选择**

EEVTEDG		边沿
0	0	无
0	1	上升沿
1	0	下降沿
1	1	任意沿

- **EEVT: 外部事件选择**

EEVT		信号选择外部事件	TIOB 方向
0	0	TIOB	输入 (1)
0	1	XC0	输出
1	0	XC1	输出
1	1	XC2	输出

Note: 1. *若选择 TIOB 作为外部事件信号，将其配置为输入并不再产生波形。*

- **ENETRG: 外部事件触发使能**

0 = 外部事件对计数器及其时钟无效。此时，所选外部事件仅控制 TIOA 输出。

1 = 外部事件复位计数器并启动计数器时钟。

- **WAVSEL: 波形选择**

WAVSEL		效果
0	0	UP 模式，无 RC 比较自动触发
1	0	UP 模式，有 RC 比较自动触发
0	1	UPDOWN 模式，无 RC 比较自动触发
1	1	UPDOWN 模式，有 RC 比较自动触发

- **WAVE = 1**

0 = 波形模式禁用 (捕获模式使能)。

1 = 波形模式使能。

- **ACPA: TIOA 上 RA 比较效果**

ACPA		效果
0	0	无
0	1	置位
1	0	清零
1	1	切换

- **ACPC: TIOA 上 RC 比较效果**

ACPC		效果
0	0	无
0	1	置位
1	0	清零
1	1	切换

- **AEEVT: TIOA 上外部事件效果**

AEEVT		效果
0	0	无
0	1	置位
1	0	清零
1	1	切换

- **ASWTRG: TIOA 上软件触发效果**

ASWTRG		效果
0	0	无
0	1	置位
1	0	清零
1	1	切换

- **BCPB: TIOB 上 RB 比较效果**

BCPB		效果
0	0	无
0	1	置位
1	0	清零
1	1	切换

- **BCPC: TIOB 上 RC 比较效果**

BCPC		效果
0	0	无
0	1	置位
1	0	清零
1	1	切换

- **BEEVT: TIOB 上外部事件效果**

BEEVT		效果
0	0	无
0	1	置位
1	0	清零
1	1	切换

- **BSWTRG: TIOB 上软件触发效果**

BSWTRG		效果
0	0	无
0	1	置位
1	0	清零
1	1	切换

TC 计数器值寄存器

寄存器名称： TC_CV

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
CV							
7	6	5	4	3	2	1	0
CV							

- **CV: 计数器值**

CV 中为实时计数器值。

TC 寄存器 A

寄存器名称： TC_RA

访问类型： 若 WAVE = 0，只读；若 WAVE = 1，读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
RA							
7	6	5	4	3	2	1	0
RA							

- **RA: 寄存器 A**

RA 中为实时寄存器 A 值。

TC 寄存器 B

寄存器名称 : TC_RB

访问类型 : 若 WAVE = 0，只读；若 WAVE = 1，读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
RB							
7	6	5	4	3	2	1	0
RB							

- **RB: 寄存器 B**

RB 中为实时寄存器 B 值。

TC 寄存器 C

寄存器名称 : TC_RC

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
RC							
7	6	5	4	3	2	1	0
RC							

- **RC: 寄存器 C**

RC 中为实时寄存器 C 值。

TC 状态寄存器

寄存器名称： TC_SR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	MTIOB	MTIOA	CLKSTA
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: 计数器溢出状态**

0 = 上次读状态寄存器后未出现计数器溢出。

1 = 上次读状态寄存器后出现计数器溢出。

- **LOVRS: 载入溢出状态**

0 = 上次读状态寄存器后未出现载入溢出，或 WAVE = 1。

1 = 若 WAVE = 0，上次读状态寄存器后，在未读取相应寄存器的情况下， RA 或 RB 已至少载入两次。

- **CPAS: RA 比较状态**

0 = 上次读状态寄存器后未出现 RA 比较，或 WAVE = 0。

1 = 若 WAVE = 1，上次读状态寄存器后出现 RA 比较。

- **CPBS: RB 比较状态**

0 = 上次读状态寄存器后未出现 RB 比较，或 WAVE = 0。

1 = 若 WAVE = 1，上次读状态寄存器后出现 RB 比较。

- **CPCS: RC 比较状态**

0 = 上次读状态寄存器后未出现 RC 比较。

1 = 上次读状态寄存器后出现 RC 比较。

- **LDRAS: RA 载入状态**

0 = 上次读状态寄存器后未出现 RA 载入，或 WAVE = 1。

1 = 若 WAVE = 0，上次读状态寄存器后出现 RA 载入。

- **LDRBS: RB 载入状态**

0 = 上次读状态寄存器后未出现 RB 载入，或 WAVE = 1。

1 = 若 WAVE = 0，上次读状态寄存器后出现 RB 载入。

- **ETRGS: 外部触发状态**

0 = 上次读状态寄存器后未出现外部触发。

1 = 上次读状态寄存器后出现外部触发。

- **CLKSTA: 时钟使能状态**

0 = 时钟禁用。

1 = 时钟使能。

- **MTIOA: TIOA 镜像**

0 = TIOA 为低。若 WAVE = 0，表示 TIOA 为低；若 WAVE = 1，表示将 TIOA 拉低。

1 = TIOA 为高，若 WAVE = 0，表示 TIOA 为高；若 WAVE = 1，表示将 TIOA 拉高。

- **MTIOB: TIOB 镜像**

0 = TIOB 为低。若 WAVE = 0，表示 TIOB 为低；若 WAVE = 1，表示将 TIOB 拉低。
1 = TIOB 为高，若 WAVE = 0，表示 TIOB 为高；若 WAVE = 1，表示将 TIOB 拉高。

TC 中断使能寄存器

寄存器名称 : TC_IER

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: 计数器溢出**

0 = 无效。

1 = 使能计数器溢出中断。

- **LOVRS: 载入溢出**

0 = 无效。

1 = 使能载入溢出中断。

- **CPAS: RA 比较**

0 = 无效。

1 = 使能 RA 比较中断。

- **CPBS: RB 比较**

0 = 无效。

1 = 使能 RB 比较中断。

- **CPCS: RC 比较**

0 = 无效。

1 = 使能 RC 比较中断。

- **LDRAS: RA 载入**

0 = 无效。

1 = 使能 RA 载入中断。

- **LDRBS: RB 载入**

0 = 无效。

1 = 使能 RB 载入中断。

- **ETRGS: 外部触发**

0 = 无效。

1 = 使能外部触发中断。

TC 中断禁用寄存器

寄存器名称 : TC_IDR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: 计数器溢出**

0 = 无效。

1 = 禁用计数器溢出中断。

- **LOVRS: 载入溢出**

0 = 无效。

1 = 禁用载入溢出中断 (若 WAVE = 0)。

- **CPAS: RA 比较**

0 = 无效。

1 = 禁用 RA 比较中断 (若 WAVE = 1)。

- **CPBS: RB 比较**

0 = 无效。

1 = 禁用 RB 比较中断 (若 WAVE = 1)。

- **CPCS: RC 比较**

0 = 无效。

1 = 禁用 RC 比较中断。

- **LDRAS: RA 载入**

0 = 无效。

1 = 禁用 RA 载入中断 (若 WAVE = 0)。

- **LDRBS: RB 载入**

0 = 无效。

1 = 禁用 RB 载入中断 (若 WAVE = 0)。

- **ETRGS: 外部触发**

0 = 无效。

1 = 禁用外部触发中断。

TC 中断屏蔽寄存器

寄存器名称： TC_IMR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

- **COVFS: 计数器溢出**

0 = 计数器溢出中断禁用。

1 = 计数器溢出中断使能。

- **LOVRS: 载入溢出**

0 = 载入溢出中断禁用。

1 = 载入溢出中断使能。

- **CPAS: RA 比较**

0 = RA 比较中断禁用。

1 = RA 比较中断使能。

- **CPBS: RB 比较**

0 = RB 比较中断禁用。

1 = RB 比较中断使能。

- **CPCS: RC 比较**

0 = RC 比较中断禁用。

1 = RC 比较中断使能。

- **LDRAS: RA 载入**

0 = 载入 RA 中断禁用。

1 = 载入 RA 中断使能。

- **LDRBS: RB 载入**

0 = 载入 RB 中断禁用。

1 = 载入 RB 中断使能。

- **ETRGS: 外部触发**

0 = 外部触发中断禁用。

1 = 外部触发中断使能。

脉宽调制控制器 (PWM)

概述

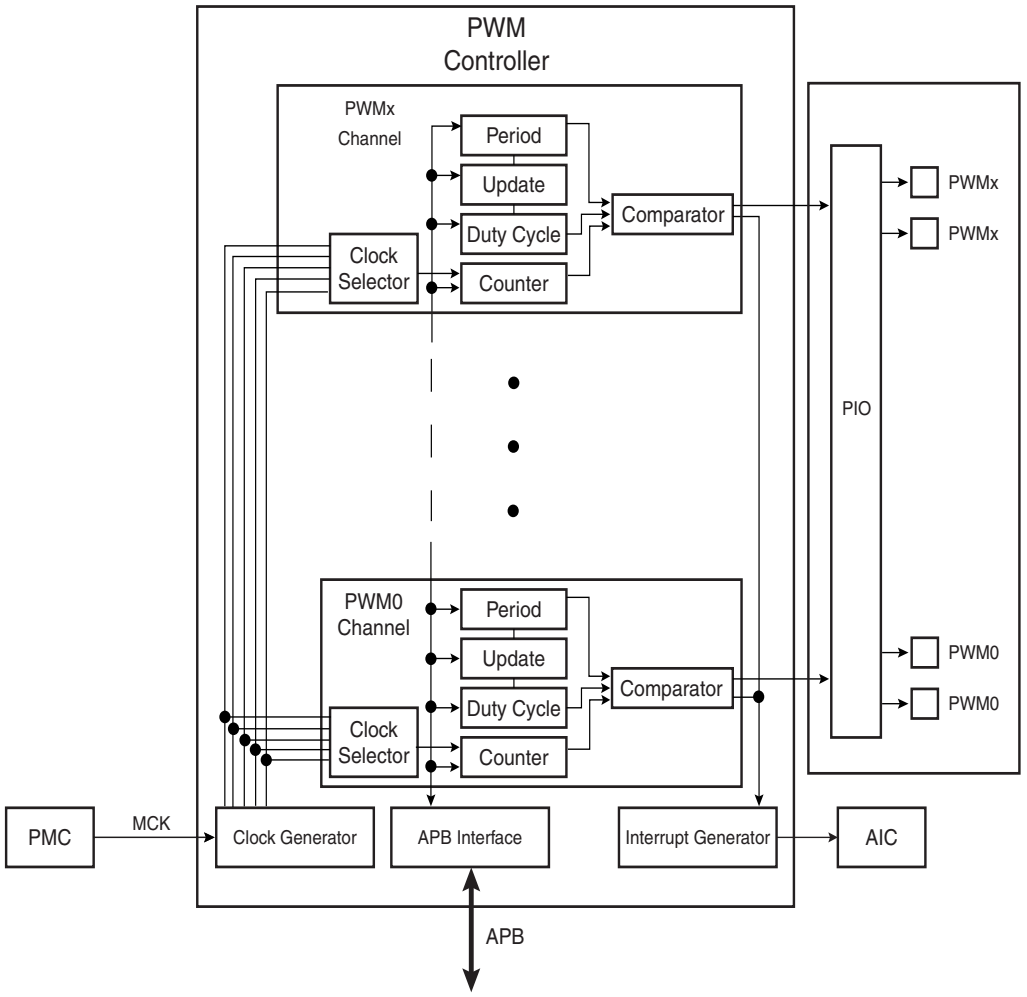
PWM 宏单元独立控制几个通道。每个通道控制一个输出方波。通过用户接口可配置输出波形的周期、占空比及极性。每个通道选择并使用一个由时钟产生器提供的时钟。时钟发生器提供几个由 PWM 宏单元分频主时钟后得到的时钟。

所有 PWM 宏单元访问通过 APB 映射寄存器实现。

可将通道同步，以产生非交迭波形。所有通道集成双缓冲系统以防止修改周期或占空比时的非预期输出波形。

方框图

Figure 162. 脉宽调制控制器框图



I/O 线说明

每个通道在每个外设 I/O 线上输出一个波形。

Table 78. I/O 线说明

名称	说明	类型
PWMx	某通道 PWM 波形输出	输出

附属产品

I/O 线

用于连接 PWM 的引脚可与 PIO 线复用。必须先编程 PIO 控制器将 PWM 引脚分配到期望的外设功能上。若应用中未使用 PWM 的 I/O 线，可由 PIO 控制器将其用作其它功能。

所有 PWM 输出可使能。若实际应用仅需要四个通道，则只为 PWM 输出分配四个 PIO 线。

电源管理

PWM 为非连续时钟。在使用 PWM 前，必须先使能电源管理控制器 (PMC) 的 PWM 时钟。但若实际应用不需要 PWM 操作，PWM 时钟可停止并在需要时重新启动。此时，PWM 将恢复到其停止前的状态。

配置 PWM 时不需要使能 PWM 时钟。

中断源

PWM 中断线与高级中断控制器的某个内部源连接。使用 PWM 中断请求前要先对 AIC 编程。注意，在边沿敏感模式下，建议不要使用 PWM 中断。

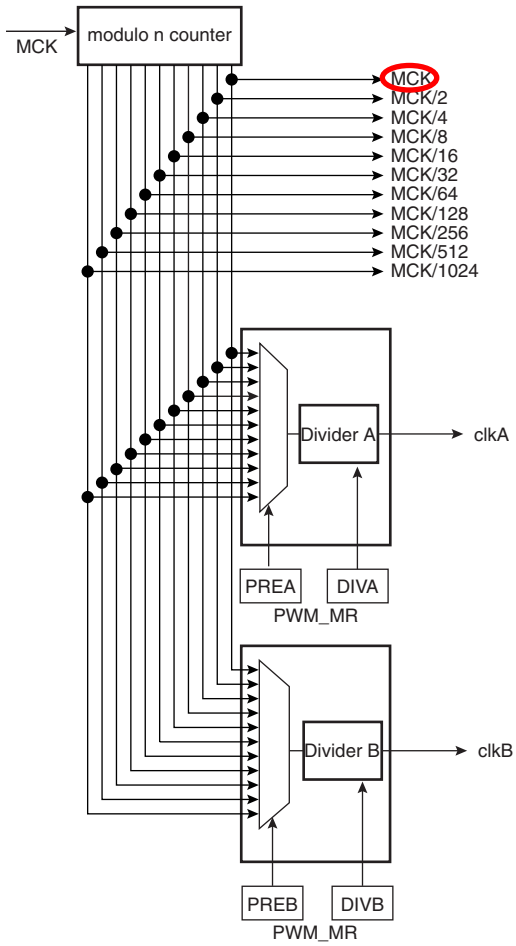
功能说明

PWM 宏单元主要由一个时钟发生器模块及 4 个通道组成。

- 由系统时钟 MCK 定时，时钟发生器模块提供 13 个时钟。
- 每个通道可独立选择一个时钟发生器输出。
- 每个通道产生的输出波形可由对应通道的用户接口寄存器独立定义。

PWM 时钟发生器

Figure 163. 时钟发生器功能图



警告：使用使用 PWM 宏单元前，必须先使能电源管理控制器 (PMC) 中的 PWM 时钟。

PWM 宏单元主机时钟 MCK 由时钟发生器模块分频后向各通道提供时钟。各通道可独立选择分频时钟：

时钟发生器在三块中分频：

- 模 n 计数器提供 11 个时钟：F_{MCK}、$F_{MCK}/2$、$F_{MCK}/4$、$F_{MCK}/8$、$F_{MCK}/16$、$F_{MCK}/32$、$F_{MCK}/64$、$F_{MCK}/128$、$F_{MCK}/256$、$F_{MCK}/512$、$F_{MCK}/1024$。
- 两个线形分频器 (1、 1/2、 1/3、 ... 1/255) 提供两个独立时钟：clkA 与 clkB。

每个线形分频器可独立对模 n 计数器时钟分频。通过 PWM 模式寄存器 (PWM_MR) 的 PREA (PREB)域选择时钟分频因子。clkA (clkB)时钟为由选定的PWM_MR寄存器DIVA (DIVB)域分频值分频后的结果。

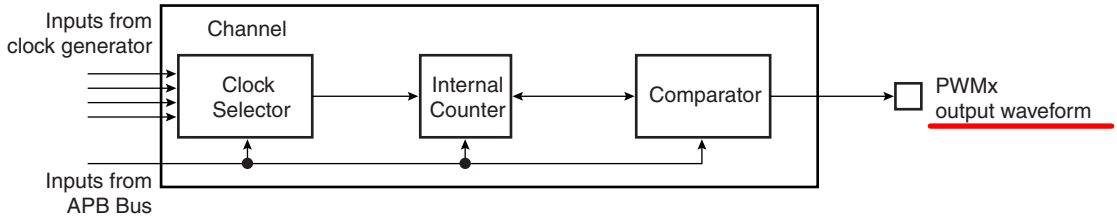
PWM控制器复位后，PWM模式寄存器中的DIVA (DIVB)及PREA (PREB)值复位为0。即复位后clkA (clkB) 关闭。

复位后，除 “clk” 外，所有由模 n 计数器提供的时钟关闭。当通过电源管理控制器将 PWM 主机时钟关闭后结果相同。

PWM 通道

方框图

Figure 164. 通道功能框图



每条通道由三部分组成：

- 选择由时钟发生器提供时钟的时钟选择器，见 “PWM 时钟发生器 ” on page 377。
- 由时钟选择器输出定时的内部计数器。根据通道配置不同及比较器事件将内部计数器递增或递减。内部计数器为 16 位。
- 比较器根据内部计数器值产生事件。它还会根据配置计算 PWMx 输出波形。

波形特性

输出波形有不同特性：

- ***内部时钟选择***。内部通道计数器由时钟发生器提供的时钟定时，见前文。该通道参数在 PWM_CMRx 寄存器 CPRE 域中定义。该域复位为 0。
- ***波形周期***。该通道参数在 PWM_CMRx 寄存器 CPRD 域中定义。
  - 若波形为左对齐，输出波形周期由计数器源时钟确定，并可计算得到：
  主机时钟 (MCK) 由给定的分频因子 X 分频
  (X 可为 1、 2、 4、 8、 16、 32、 64、 128、 256、 512 或 1024)，公式为：

  $$\frac{(X \times CPRD)}{MCK}$$

  主机时钟由 DIVA 及 DIVB 分频器单独或共同分频，公式为：

  $$\frac{(CRPD \times DIVA)}{MCK} \text{ 或 } \frac{(CRPD \times DIVAB)}{MCK}$$

  若波形为中心对齐则输出波形周期由计数器源时钟确定，并可计算得到：
  主机时钟 (MCK) 由给定的分频因子 X 分频
  (X 可为 1、 2、 4、 8、 16、 32、 64、 128、 256、 512 或 1024)，公式为：

  $$\frac{(2 \times X \times CPRD)}{MCK}$$

  主机时钟由 DIVA 及 DIVB 分频器单独或共同分频，公式为：

  $$\frac{(2 \times CPRD \times DIVA)}{MCK} \text{ 或 } \frac{(2 \times CPRD \times DIVB)}{MCK}$$

- ***波形占空比***。该通道参数在 PWM_CDTYx 寄存器 CDTY 域中定义。
  若波形为左对齐：

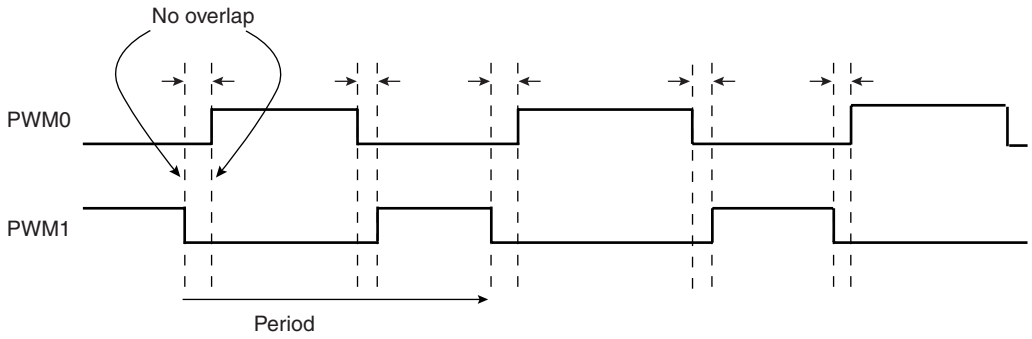
  $$\text{duty cycle} = \frac{(period - 1 / \text{ fchannel\_x\_clock} \times CDTY)}{period}$$

  若波形为中心对齐：

$$\text{uty cycle} = \frac{((period/\ 2) - 1/\ \text{fchannel\_x\_clock} \times CDTY)}{(period/\ 2)}$$

- ***波形极性***。周期开始时 ,信号可为任意电平。该特性在 PWM_CMRx 寄存器的 CPOL 域中定义。默认信号以低电平启动。
- ***波形对齐***。输出波形可左对齐或中心对齐。中心对齐波形可用来产生无交迭波形。该特性在 PWM_CMRx 寄存器的 CALG 域定义。默认为左对齐。

Figure 165. 无交迭中心对齐波形



Note: 1. 中心对齐波形说明详见 Figure 166 on page 380。

当中心对齐时，内部计数器递增到 CPRD 然后递减到 0，结束周期。

当左对齐时，内部计数器递增到 CPRD 然后复位，结束周期。

因此，等于相同的 CPRD 值，中心对齐通道是左对齐通道周期的两倍。

波形固定为 0，当：

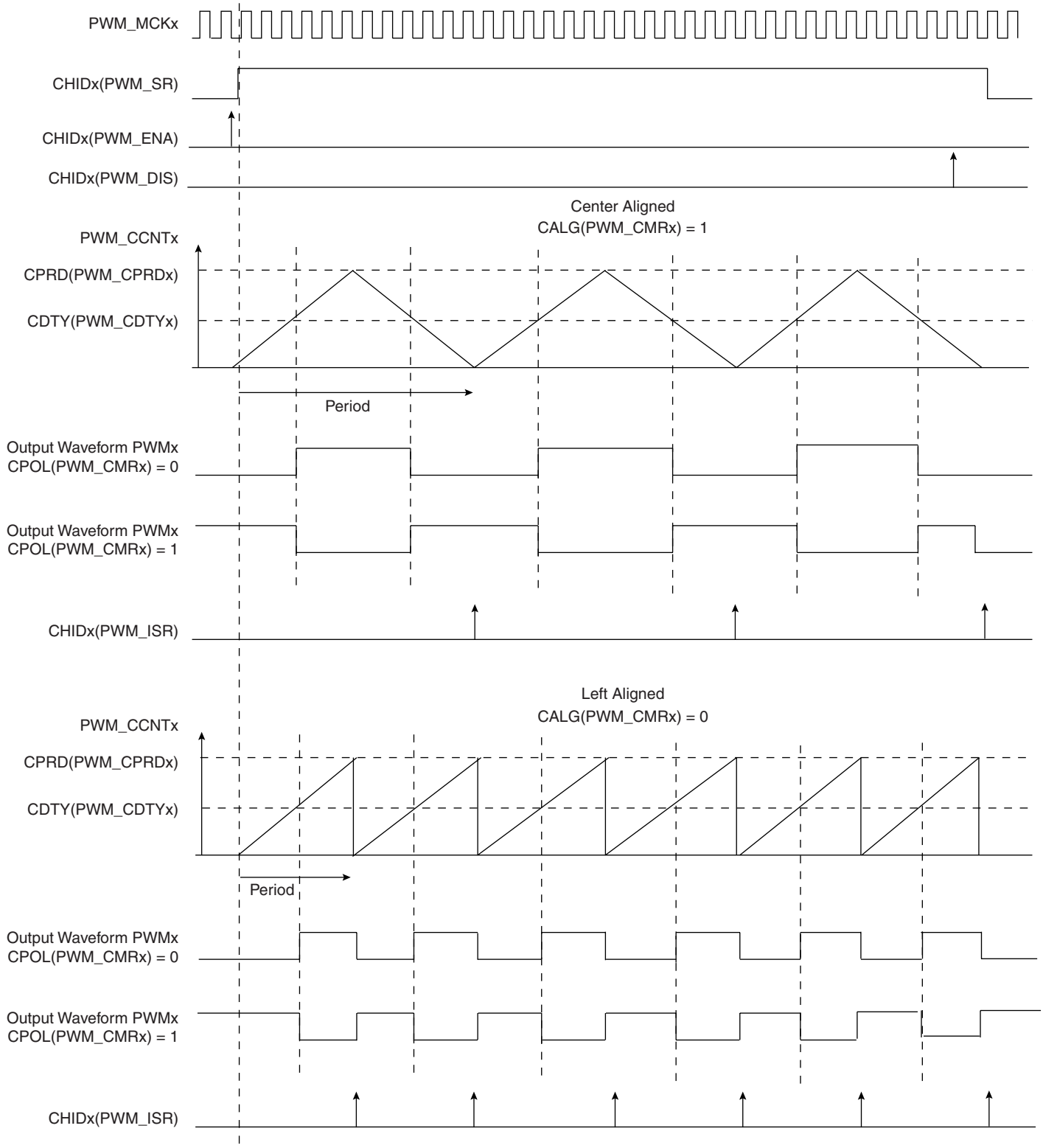
- CDTY = CPRD 且 CPOL = 0
- CDTY = 0 且 CPOL = 1

波形固定为 1 (一旦通道使能)，当：

- CDTY = 0 且 CPOL = 0
- CDTY = CPRD 且 CPOL = 1

使能通道前必须设置波形极性。这会立即影响通道输出电平。通道使能时，不会记录通道极性改变。

Figure 166. 波形特性



PWM 控制器操作

初始化

使能输出通道前，该通道必须由软件配置：

- 若需要 DIVA 与 DIVB，配置时钟发生器。
- 为每条通道选择时钟 (PWM_CMRx 寄存器中的 CPRE 域)。
- 为每条通道配置波形对齐 (PWM_CMRx 寄存器中的 CALG 域)。
- 为每条通道配置周期 (PWM_CPRDx 寄存器中的 CPRD 域)。通道禁时可以写入 PWM_CPRDx 寄存器。通道有效后，必须使用 PWM_CUPDx 寄存器更新 PWM_CPRDx，说明见后。
- 为每条通道配置占空比 (PWM_CDTYx 寄存器中的 CDTY 域)。通道禁时可以写入 PWM_CDTYx 寄存器。通道有效后，必须使用 PWM_CUPDx 寄存器更新 PWM_CDTYx，说明见后。
- 为每条通道配置输出波形极性 (PWM_CMRx 寄存器中的 CPOL 域)。
- 使能中断 (写 PWM_IER 寄存器中的 CHIDx)。
- 使能 PWM 通道 (写 PWM_ENA 寄存器中的 CHIDx)。

同时在 PWM_ENA 寄存器中对几个 CHIDx 位写入将同步不同通道。

- 此时，所有通道时钟选择器配置相同，周期相同。

源时钟选择标准

时钟源有多种选择。周期寄存器 (PWM_CPRDx) 与占空比寄存器 (PWM_CDTYx) 值间的关系可帮助选择。写入周期寄存器的事件数给出 PWM 精度。占空比不能低于 *1/PWM_CPRDx*。PWM_CPRDx 值越高，PWM 精度越高。

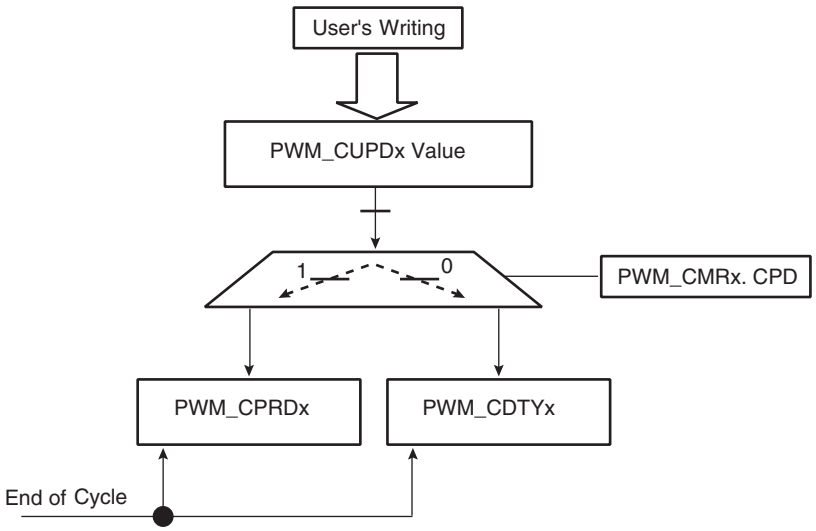
例如，若 PWM_CPRDx 为 15 (十进制)，用户可在 PWM_CDTYx 寄存器中设置 1 到 14 间值。占空比不能低于 PWM 周期的 1/15。

改变占空比或周期

输出波形占空比或周期可调。

为防止通道使能时修改波形参数出现非预期波形，将对 PWM_CPRDx 及 PWM_CDTYx 寄存器进行双缓冲。用户可在更新寄存器 (PWM_CUPDx) 中写入新周期或占空比值。该寄存器将新值保持到当前周期结束并在下一个周期更新。根据 PWM_CMRx 寄存器 CPD 域的不同，PWM_CUPDx 更新 PWM_CPRDx 或 PWM_CDTYx。

Figure 167. 周期同步或占空比更新



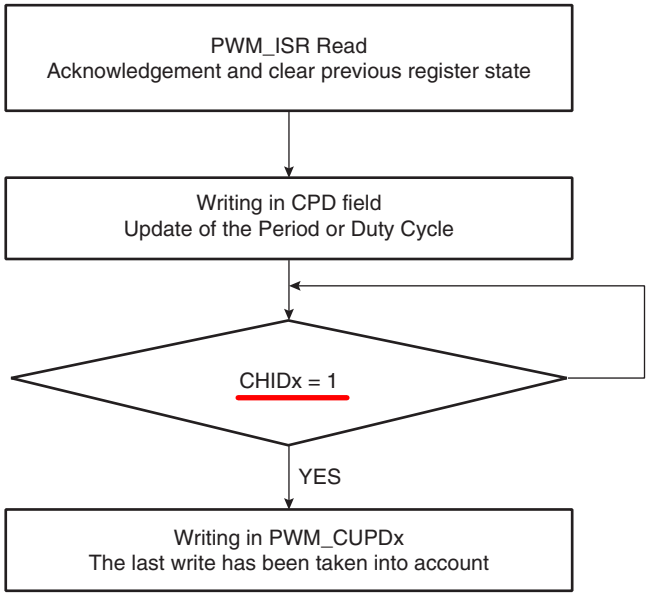
为防止软件覆盖 PWM_CUPDx，用户可使用状态事件来同步其软件。两种方法均可行。两种情况下，用户均须将 PWM_IER 专用中断使能到 PWM 控制器级。

第一种方式 (轮询方式) 会根据使能的通道读 PWM_ISR 寄存器中相关位，见 Figure 168。

第二种方式使用与 PWM 通道相关的中断服务子程序。

Note: 读 PWM_ISR 寄存器将自动清除 CHIDx 标志。

Figure 168. 轮询方式



PWM 用户接口

PWM 寄存器映射

Table 79. PWM 控制器寄存器

偏移	寄存器	名称	访问类型	外设复位值
0x00	PWM 模式寄存器	PWM_MR	读 / 写	0
0x04	PWM 使能寄存器	PWM_ENA	只写	-
0x08	PWM 禁用寄存器	PWM_DIS	只写	-
0x0C	PWM 状态寄存器	PWM_SR	只读	0
0x10	PWM 中断使能寄存器	PWM_IER	只写	-
0x14	PWM 中断禁用寄存器	PWM_IDR	只写	-
0x18	PWM 中断屏蔽寄存器	PWM_IMR	只读	0
0x1C	PWM 中断状态寄存器	PWM_ISR	只读	0
0x4C - 0xF8	保留	–	–	–
0xFC	版本寄存器	PWM_VERSION	只读	0x- (1)
0x100 - 0x1FC	保留			
0x200	通道 0 模式寄存器	PWM_CMR0	读 / 写	0x0
0x204	通道 0 占空比寄存器	PWM_CDTY0	读 / 写	0x0
0x208	通道 0 周期寄存器	PWM_CPRD0	读 / 写	0x0
0x20C	通道 0 计数器寄存器	PWM_CCNT0	只读	0x0
0x210	通道 0 更新寄存器	PWM_CUPD0	只写	-
...	保留			
0x220	通道 1 模式寄存器	PWM_CMR1	读 / 写	0x0
0x224	通道 1 占空比寄存器	PWM_CDTY1	读 / 写	0x0
0x228	通道 1 周期寄存器	PWM_CPRD1	读 / 写	0x0
0x22C	通道 1 计数器寄存器	PWM_CCNT1	只读	0x0
0x230	通道 1 更新寄存器	PWM_CUPD1	只写	-
...	...	...	...	...

Note: 1. 版本寄存器中值根据 IP 块版本不同而不同。

PWM 模式寄存器

寄存器名称： PWM_MR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	PREB			

23	22	21	20	19	18	17	16
DIVB							

15	14	13	12	11	10	9	8
–	–	–	–	PREA			

7	6	5	4	3	2	1	0
DIVA							

- **DIVA, DIVB: CLKA, CLKB 分频因子**

DIVA, DIVB	CLKA, CLKB
0	CLKA、 CLKB 时钟关闭
1	CLKA、 CLKB 时钟由 PREA、 PREB 选定
2-255	CLKA、 CLKB 时钟由 PREA、 PREB 由 DIVA、 DIVB 因子分频后得到

- **PREA, PREB**

PREA, PREB				分频器输入时钟
0	0	0	0	MCK.
0	0	0	1	MCK/2
0	0	1	0	MCK/4
0	0	1	1	MCK/8
0	1	0	0	MCK/16
0	1	0	1	MCK/32
0	1	1	0	MCK/64
0	1	1	1	MCK/128
1	0	0	0	MCK/256
1	0	0	1	MCK/512
1	0	1	0	MCK/1024
其它				保留

PWM 使能寄存器

寄存器名称 : PWM_ENA

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: 通道 ID**

0 = 无效。

1 = 使能通道 x 的 PWM 输出。

PWM 禁用寄存器

寄存器名称 : PWM_DIS

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: 通道 ID**

0 = 无效。

1 = 禁用通道 x 的 PWM 输出。

PWM 状态寄存器

寄存器名称： PWM_SR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: 通道 ID**

0 = 通道 x PWM 输出禁用。

1 = 通道 x PWM 输出使能。

PWM 中断使能寄存器

寄存器名称： PWM_IER

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: 通道 ID**

0 = 无效。

1 = 使能通道 x PWM 中断。

PWM 中断禁用寄存器

寄存器名称： PWM_IDR

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: 通道 ID**

0 = 无效。

1 = 禁用通道 x PWM 中断。

PWM 中断屏蔽寄存器

寄存器名称： PWM_IMR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: 通道 ID**

0 = 通道 x PWM 中断禁用。

1 = 通道 x PWM 中断使能。

PWM 中断状态寄存器

寄存器名称： PWM_ISR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–	–	–	CHID3	CHID2	CHID1	CHID0

- **CHIDx: 通道 ID**

0 = 上次读 PWM_ISR 寄存器后无新通道周期实现。

1 = 上次读 PWM_ISR 寄存器后至少有一个通道周期实现。

注意：读 PWM_ISR 将自动清除 CHIDx 标志。

PWM 通道模式寄存器

寄存器名称 : PWM_CMRx

A 访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	CPD	CPOL	CALG

7	6	5	4	3	2	1	0
–	–	–	–	CPRE			

- **CPRE: 通道预分频**

CPRE				通道预分频
0	0	0	0	MCK
0	0	0	1	MCK/2
0	0	1	0	MCK/4
0	0	1	1	MCK/8
0	1	0	0	MCK/16
0	1	0	1	MCK/32
0	1	1	0	MCK/64
0	1	1	1	MCK/128
1	0	0	0	MCK/256
1	0	0	1	MCK/512
1	0	1	0	MCK/1024
1	0	1	1	CLKA
1	1	0	0	CLKB
其它				保留

- **CALG: 通道对齐**

0 = 周期左对齐。

1 = 周期中心对齐。

- **CPOL: 通道极性**

0 = 输出波形在低电平启动。

1 = 输出波形在高电平启动。

- **CPD: 通道更新周期**

0 = 写 PWM_CUPDx 将在下一个周期开始时修改占空比。

1 = 写 PWM_CUPDx 将在下一个周期开始时修改周期。

PWM 通道占空比寄存器

寄存器名称 : PWM_CDTYx

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
CDTY							

23	22	21	20	19	18	17	16
CDTY							

15	14	13	12	11	10	9	8
CDTY							

7	6	5	4	3	2	1	0
CDTY							

只有前 16 位 (内部通道寄存器大小) 有意义。

- **CDTY: 通道占空比**

定义波形占空比。该值必须定义在 0 与 CPRD (PWM_CPRx) 间。

PWM 通道周期寄存器

寄存器名称： PWM_CPRDx

访问类型： 读 / 写

31	30	29	28	27	26	25	24
CPRD							
23	22	21	20	19	18	17	16
CPRD							
15	14	13	12	11	10	9	8
CPRD							
7	6	5	4	3	2	1	0
CPRD							

只有前 16 位 (内部通道寄存器大小) 有意义。

- **CPRD: 通道周期**

若波形为左对齐，输出波形周期由计数器源时钟确定，并可计算得到：

– 主机时钟 (MCK) 由给定的分频因子 X 分频 (X 可为 1、2、4、8、16、32、64、128、256、512 或 1024)，公式为：

$$\frac{(X \times CPRD)}{MCK}$$

– 主机时钟由 DIVA 或 DIVB 分频器分频，公式为：

$$\frac{(CRPD \times DIVA)}{MCK} \text{ 或 } \frac{(CRPD \times DIVAB)}{MCK}$$

若波形为中心对齐，输出波形周期由计数器源时钟确定，并可计算得到：

– 主机时钟 (MCK) 由给定的分频因子 X 分频 (X 可为 1、2、4、8、16、32、64、128、256、512 或 1024)，公式为：

$$\frac{(2 \times X \times CPRD)}{MCK}$$

– 主机时钟由 DIVA 或 DIVB 分频器分频，公式为：

$$\frac{(2 \times CPRD \times DIVA)}{MCK} \text{ 或 } \frac{(2 \times CPRD \times DIVB)}{MCK}$$

PWM 通道计数寄存器

寄存器名称 : PWM_CCNTx

访问类型 : 只读

31	30	29	28	27	26	25	24
CNT							

23	22	21	20	19	18	17	16
CNT							

15	14	13	12	11	10	9	8
CNT							

7	6	5	4	3	2	1	0
CNT							

- **CNT: 通道计数器寄存器**

内部计数器值，该寄存器在下列情况时复位：

- 通道使能 (写 PWM_ENA 寄存器的 CHIDx)。
- 若波形为左对齐计数器达到 PWM_CPRDx 寄存器中定义的 CPRD 值。

PWM 通道更新寄存器

寄存器名称 : PWM_CUPDx

访问类型 : 只写

31	30	29	28	27	26	25	24
CUPD							

23	22	21	20	19	18	17	16
CUPD							

15	14	13	12	11	10	9	8
CUPD							

7	6	5	4	3	2	1	0
CUPD							

该寄存器实际上是周期或占空比的双缓冲器。防止修改波形周期或占空比时出现非期望波形。

只有前 16 位 (内部通道寄存器大小) 有意义。

CPD (PWM_CMRx Register)	
0	在下一周期开始时，占空比 (PWM_CDRx 寄存器中的 CDTC) 以 CUPD 值更新。
1	在下一周期开始时，周期 (PWM_CDRx 寄存器中的 CPRD) 以 CUPD 值更新。

PWM 版本寄存器

寄存器名称 : PWM_VERSION

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	MFN		
15	14	13	12	11	10	9	8
–	–	–	–	VERSION			
7	6	5	4	3	2	1	0
VERSION							

- **VERSION**

保留。值可变。无相关功能。是 Atmel 宏单元内部版本。

- **MFN**

保留。值可变。无相关功能。

USB 器件端口 (UDP)

概述

USB 器件端口 (UDP) 适用于通用串行总线 (USB) V2.0 全速器件规范。

每个端点可配置为几种 USB 传输类型中的一种。它可与双端 RAM 的一段或两段联合用来存储当前数据有效负载。若使用两段，一个 DPR 段由处理器读、写，另外一个 DPR 段则由 USB 器件外设读、写。对于同步端点强制使用该特性。因此器件工作于有两 DPR 段端点时，保持最大带宽 (1M 字节 /s)。

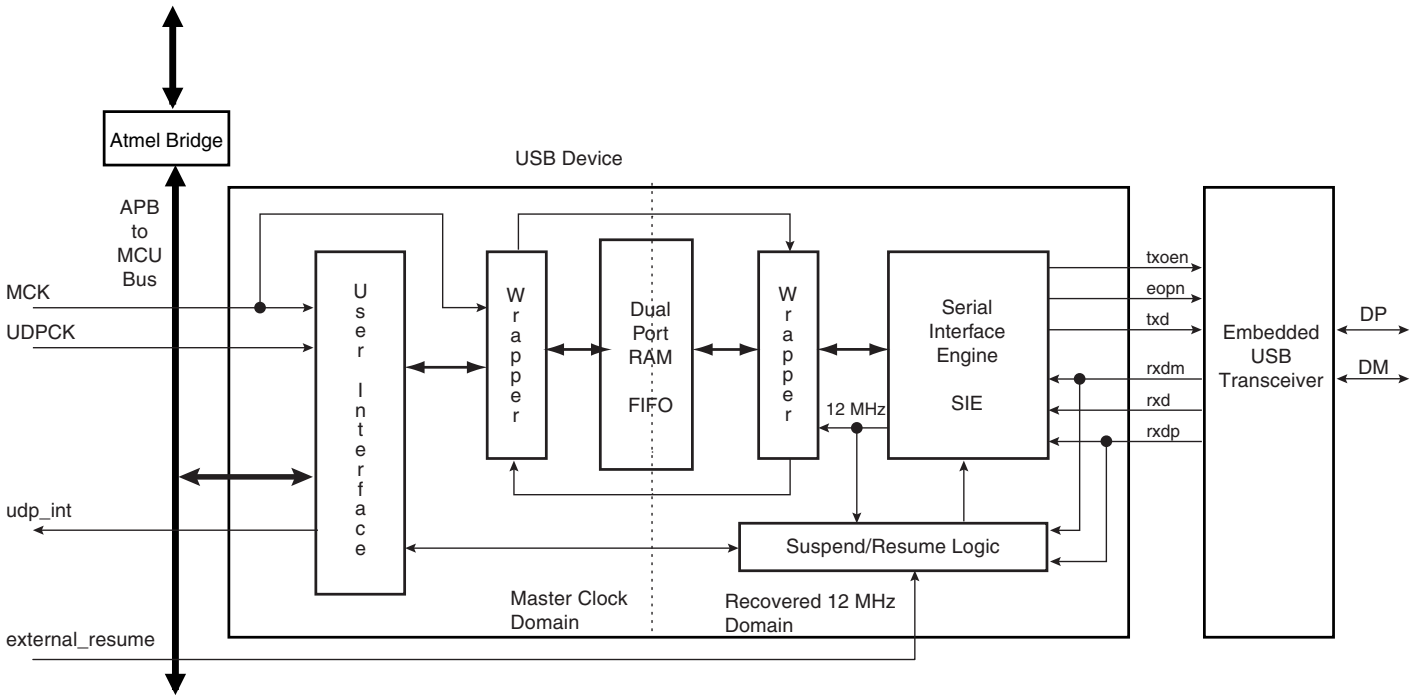
Table 80. USB 端点说明

端点序号	助记符	双段	最大端点大小	端点类型
0	EP0	非	8	控制 / 批 / 中断
1	EP1	是	64	批 / 单独 / 中断
3	EP2	是	64	批 / 单独 / 中断
3	EP3	非	64	控制 / 批 / 中断

USB 器件自动检测挂起与恢复，通过出现中断来停止处理器。某些产品中可利用外部信号唤醒 USB 主机控制器发送。

方框图

Figure 169. 方框图



通过 APB 总线接口访问 UDP。通过对 APB 寄存器 8 位值读写来对数据 FIFO 进行读写。

UDP 外设需要两个时钟：用于 MCK 域的外设时钟及用于 12 MHz 域的 48 MHz 时钟。

USB 2.0 全速垫内置并由串行接口引擎 (SIE) 控制。

external_resume信号可选。它允许在系统模式下唤醒UDP外设。然后主机将通知请求恢复的器件。列举时，该特性必须由主机处理。

附属产品

USB 器件硬件说明，详见 “USB Device Port” on page 28。

USB 物理收发器集成在产品中。双向差分信号 DP 与 DM 对于产品边界有效。

应用中可能会用到两个 I/O 线：

- 一条检查来自主机的 VBUS 是否仍然有效。可能使用该入口通知自供电器件主机断电。此时，禁用板上上拉 DP 以防止电流流入主机。
- 一条用来控制板上上拉 DP。因此，当器件准备与主机通信时，通过该控制线激活其 DP 上拉。

I/O 线

PIO 控制器不控制 DP 与 DM。内置的 USB 物理收发器由 USB 器件外设控制。

为保留检查 VBUS 的 I/O 线，必须先对 PIO 控制器编程，将该 I/O 配置为输入 PIO 模式。

为保留板上拉控制 I/O 线，必须先对 PIO 控制器编程，将该 I/O 配置为输出 PIO 模式。

电源管理

USB 器件外设需要 48 MHz 时钟。该时钟由精度为 ± 0.25% 的 PLL 产生。

因此，USB 器件收到来自电源管理控制器 (PMC) 的两个时钟：主机时钟 MCK 用来驱动外设用户接口； UDPCK 用来与总线 USB 信号连接 (恢复的 12 MHz 域)。

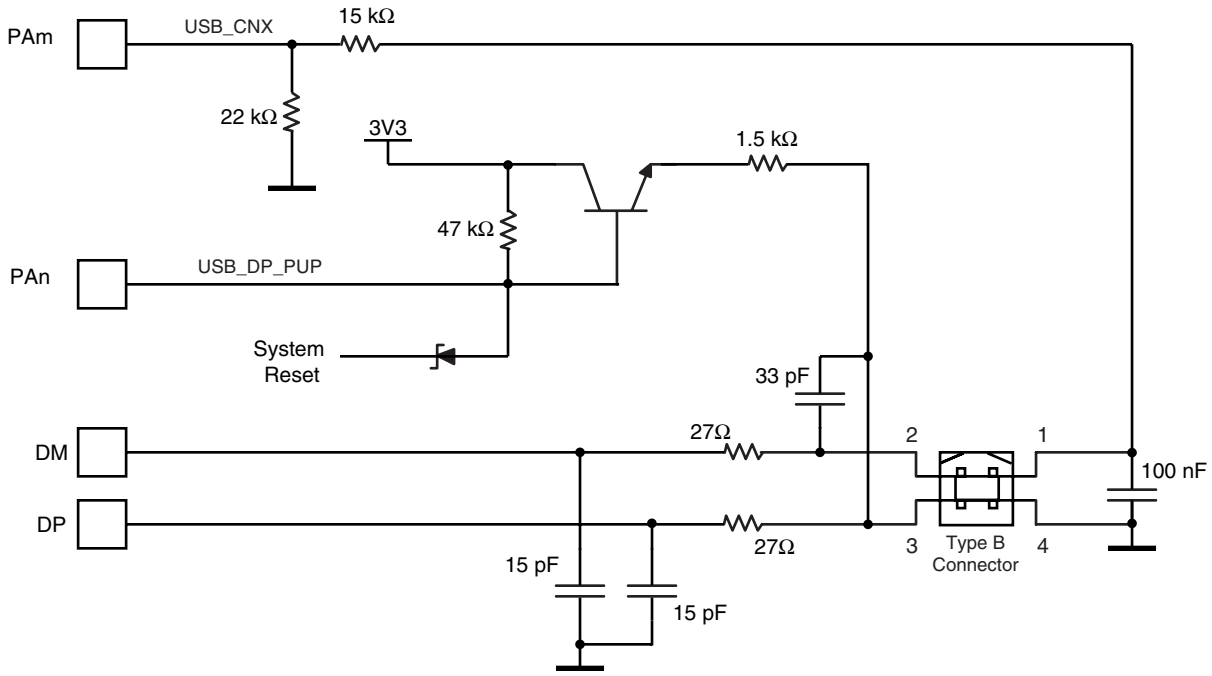
中断

USB 器件有一条与高级中断控制器 (AIC) 连接的中断线。

处理 USB 器件中断须在配置 UDP 前对 AIC 编程。

典型连接

Figure 170. 与 USB 器件外设连接示意图



输入信号 USB_CNX 用来检验主机是否连接。

输出信号 USB_DP_PUP 用来使能 DP 上拉。

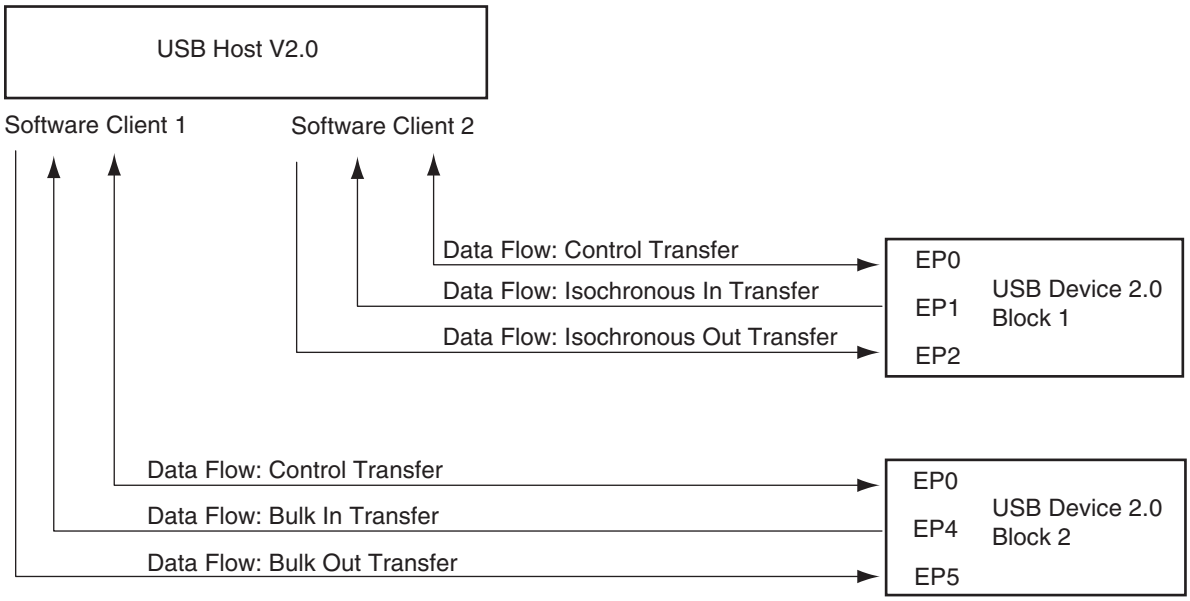
Figure 170 给出复位后上拉自动激活示意图。

功能说明

USB V2.0 全速介绍

USB V2.0 全速提供主机与附属 USB 器件间的通信服务。每个器件提供与每个端点相关的通信流。软件通过通信流实现主机与 USB 器件的通信。

Figure 171. USB V2.0 全速通信控制示例



USB V2.0 全速传输类型

USB 器件定义四种传输类型。

Table 81. USB 通信流

传输	方向	带宽	端点大小	错误检测	重试
控制	双向	无保证	8, 16, 32, 64	有	自动
同步	单向	保证	1 - 1023	有	无
中断	单向	无保证	≤64	有	有
批	单向	无保证	8, 16, 32, 64	有	有

USB 总线处理

每次发送将在 USB 总线上进行一个或多个处理。共有五种处理类型：

1. 设置处理
2. 数据入处理
3. 数据出处理
4. 状态入处理
5. 状态出处理

USB 发送事件定义

如下所示，USB 总线上发送顺序进行。

Table 82. USB 发送事件

控制发送 (1) (3)	• 设置处理 > 数据入处理 > 状态出处理 • 设置处理 > 数据出处理 > 状态入处理 • 设置处理 > 状态入处理
中断入发送 (器件向主机)	• 数据入处理 > 数据入处理
中断出发送 (主机向器件)	• 数据出处理 > 数据出处理
同步入发送 (2) (器件向主机)	• 数据入处理 > 数据入处理
同步出发送 (2) (主机向器件)	• 数据出处理 > 数据出处理
批入发送 (器件向主机)	• 数据入处理 > 数据入处理
批出发送 (主机向器件)	• 数据出处理 > 数据出处理

Notes: 1. 控制发送必须使用无 ping-pong 特性的端点。
2. 同步发送必须使用有 ping-pong 特性的端点。
3. 使用延迟握手可忽略控制发送。

与 USB V2.0 器件外设交互处理

设置处理

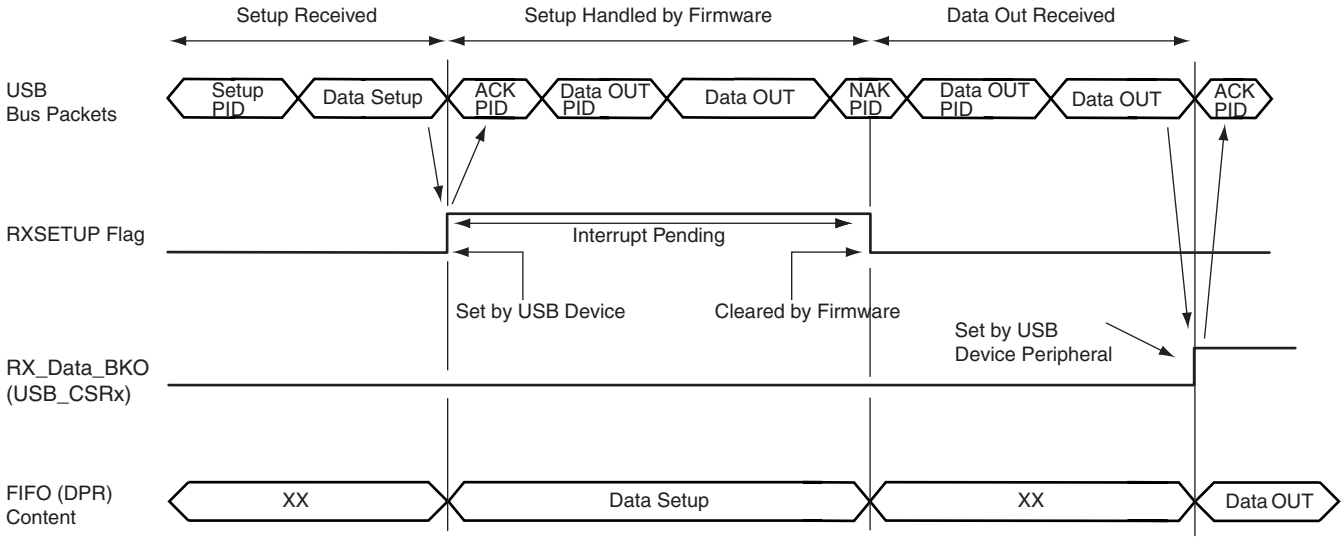
设置是控制发送时主机到器件处理的专用类型。控制发送必须使用无 ping-pong 特性的端点。设置处理应尽快由固件实现。它使用从主机到器件的发送请求。然后这些请求由 USB 器件处理，并可能需要更多的变量。变量由设置处理后的数据出处理发送到器件。这些处理也可能返回数据。这些数据由设置处理后的下一个数据入处理送入主机。状态处理结束整个控制发送。

当 USB 端点收到设置发送时：

- USB 器件自动应答该设置包。
- 设置 USB_CSRx 寄存器中的 RXSETUP。
- 若 RXSETUP 未被清零则产生一个端点中断。若该端点中断使能，该中断在微控制器上执行。

因此，固件必须轮询 USB_CSRx 以检测 RXSETUP 或获取中断，从 FIFO 中读设置包，然后清除 RXSETUP。在未读 FIFO 中数据包前，RXSETUP 不能清零。否则，USB 器件将接收下一个数据出发送并覆盖 FIFO 中设置包。

Figure 172. 设置处理及数据出处理



数据入处理

数据入处理用在控制、同步、批及中断处理中，引导数据由器件送往主机。同步发送中使用数据入处理的端点必须有 ping-pong 特性。

使用无 Ping-pong 特性端点

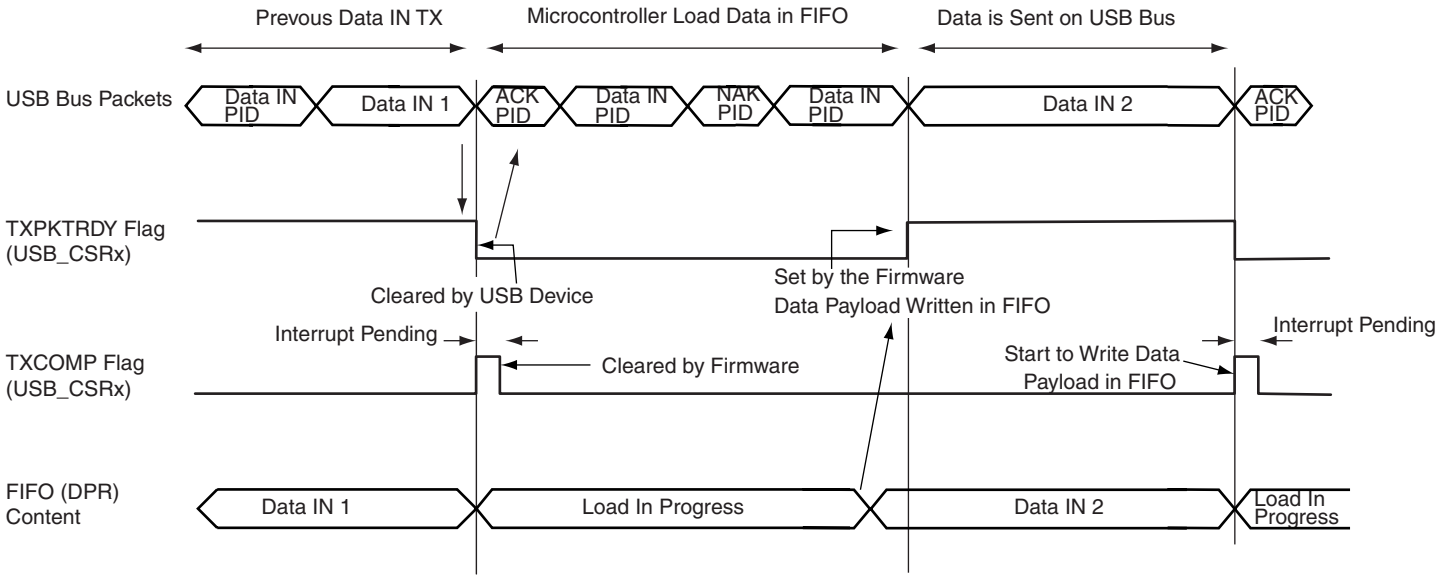
用无 ping-pong 端点执行数据入处理：

1. 微控制器检查是否通过轮询端点 USB_CSRx 寄存器的 TXPKTRDY 对 FIFO 写入 (TXPK-TRDY 必须清零)。
2. 微控制器将准备发送的数据写入端点的FIFO中，给端点USB_FDRx寄存器写入0或多字节值。
3. 微控制器通过设置端点 USB_CSRx 寄存器的 TXPKTRDY 位来通知 USB 外设它已完成。
4. 当 USB_CSRx 寄存器已置位，微控制器通知已释放端点 FIFO。然后当 TXCOMP 置位时相应端点中断挂起。

当 USB 器件收到数据入包 ACK PID 信号时，将 TXCOMP 置位。当 TXCOMP 置位时中断挂起。

Note: 更多关于数据入协议层的内容请参见*通用串行总线规范， Rev 2.0*。

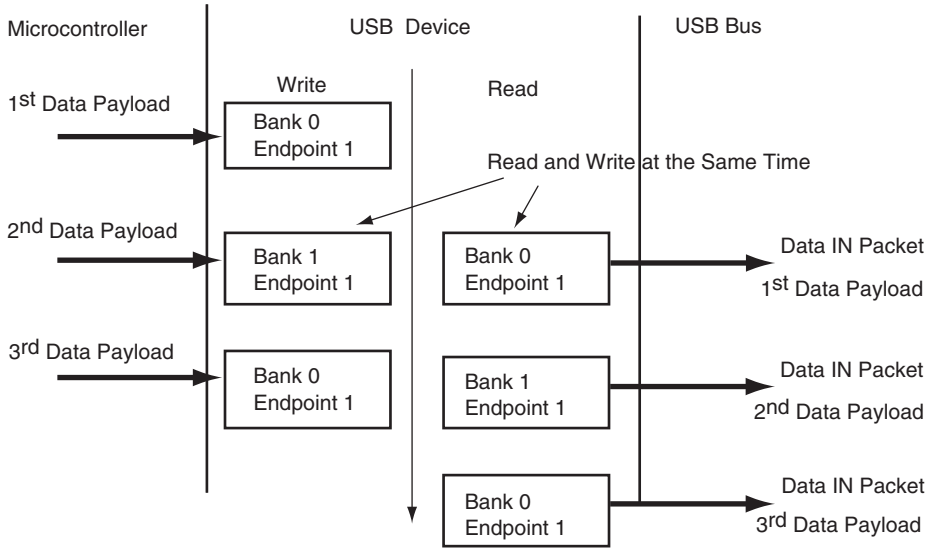
Figure 173. 无 Ping-pong 端点的数据入发送



使用有 Ping-pong 特性端点

同步发送时需要使用有 ping-pong 特性的端点。为保证带宽为常数，当 USB 器件发送当前数据时，微控制器必须准备下一个数据有效负载。因此要使用两个存储器段。一个给微控制器使用，另一个由 USB 器件锁定。

Figure 174. Ping-pong 端点数据入发送时的段交换

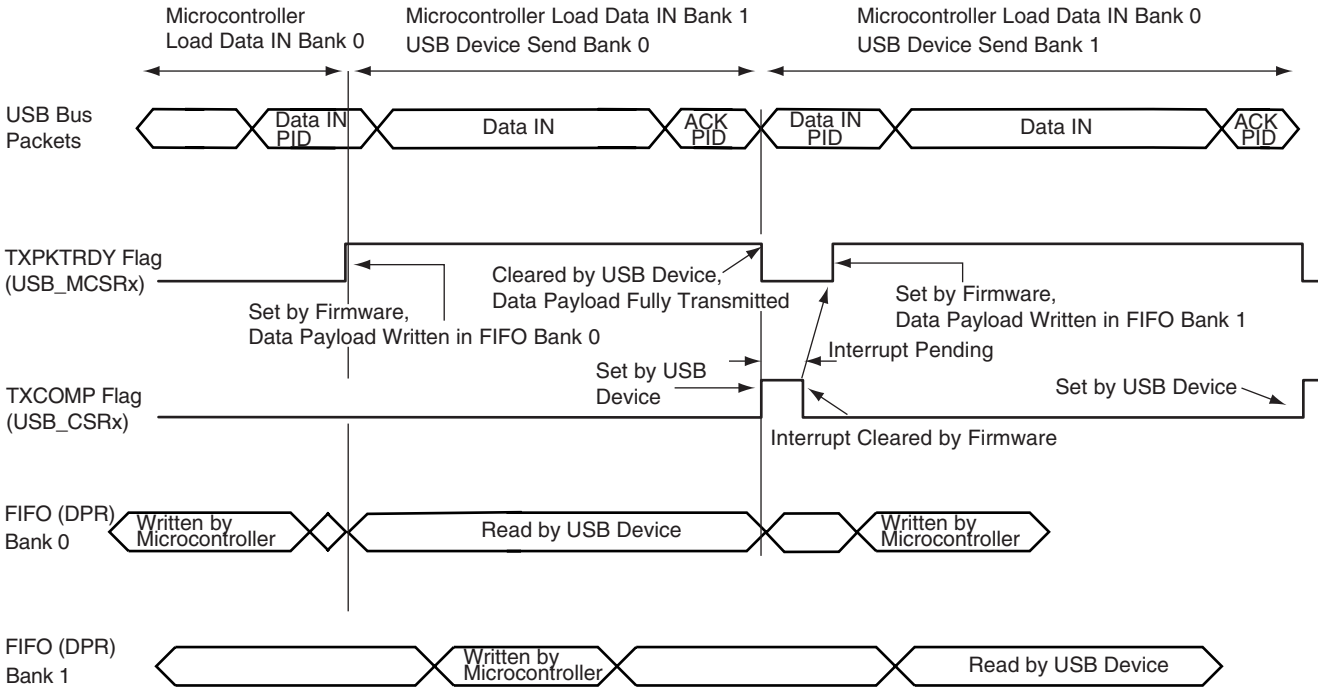


使用 ping-pong 端点时，数据入处理按以下步骤执行：

1. 微控制器通过轮询端点USB_CSRx寄存器TXPKTRDY是否清零来检查是否能在FIFO中写入。
2. 微控制器将要发送的第一个数据有效负载写入FIFO (段0)，在端点USB_FDRx寄存器写入 0 或多字节值。
3. 微控制器通过设置端点 USB_CSRx 寄存器的 TXPKTRDY 位来通知 USB 外设，它已完成对 FIFO 段 0 的写入。

4. 不必等待 TXPKTRDY 清零，微控制器将第二个数据有效负载写入 FIFO (段 1)，在端点 USB_FDRx 寄存器写入 0 或多字节值。
5. 当 USB_CSRx 寄存器的 TXCOMP 置位， USB 器件通知微控制器已经释放首个段。TXCOMP 置位时将中断挂起。
6. 一旦微控制器收到首个段的 TXCOMP 时，它将端点 USB_CSRx 寄存器的 TXPKTRDY 拉高以通知 USB 器件准备发送第二个段数据。
7. 此步中，段 0 有效，微控制器开始准备发送第三个数据有效负载。

Figure 175. Ping-pong 端点的数据入发送



警告 : 这是软件关键路径，由于一旦第二个段满，驱动器须等待 TX_COMP 设置 TX_PKTRDY。若接收 TX_COMP 置位与 TX_PKTRDY 设置间延迟时间太长，某些数据入包可能无应答，减低带宽。

数据出处理

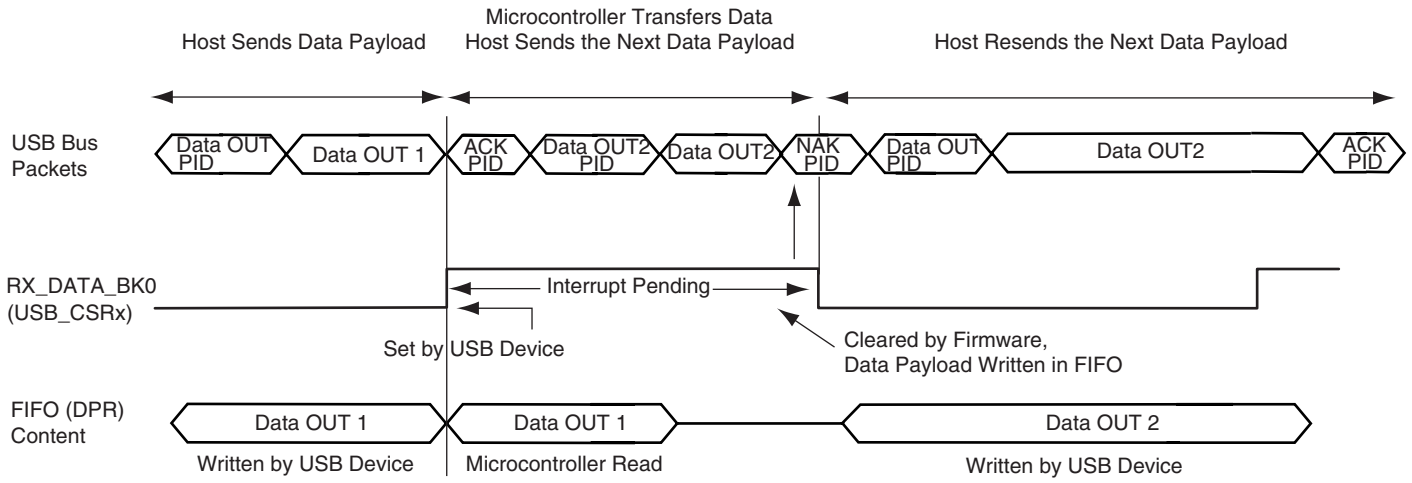
数据出处理用在控制、同步、批及中断传输中，并引导数据由主机向器件发送。同步发送中的数据出处理必须使用有 ping-pong 特性的端点中。

无 Ping-pong 特性的数据出处理

用无 ping-pong 端点执行数据出处理：

1. 主机产生数据出包。
2. USB 器件端点接收包。当与该端点相关的 FIFO 由微控制器使用时，NAK PID 信号返回主机。一旦 FIFO 有效， USB 器件将数据写入 FIFO 并自动给主机发送一个 ACK。
3. 通过轮询端点 USB_CSRx 寄存器的 RX_DATA_BK0 通知微控制器 USB 器件已接收到数据有效负载。当 RX_DATA_BK0 置位时，该端点中断挂起。
4. 读取端点 USB_CSRx 寄存器的 RXBYTECNT 得到 FIFO 中字节数目。
5. 微控制器接收端点存储器的数据到其存储器中。读端点 USB_FDRx 寄存器可得到接收数据。
6. 通过清零端点 USB_CSRx 寄存器的 RX_DATA_BK0，微控制器通知 USB 器件发送完成。
7. USB 器件可接收新数据出包。

Figure 176. 无 Ping-pong 端点数据出发送

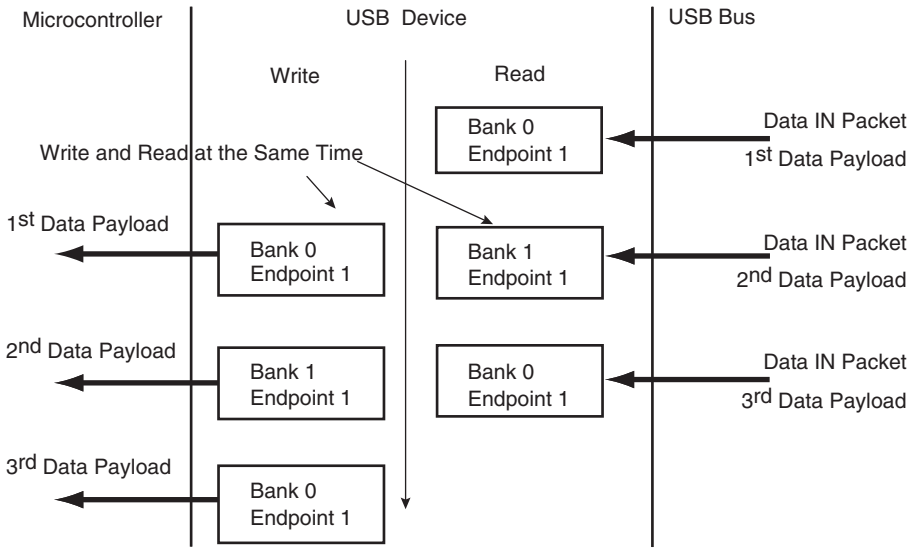


当 RX_DATA_BK0 标志置位，中断挂起。 RX_DATA_BK0 清零后， USB 器件、 FIFO 及微控制器存储器间发送不能进行。否则， USB 器件将接收下一个数据出发送并覆盖当前 FIFO 中的数据出包。

使用有 Ping-pong 特性的端点

同步发送时，必须使用有 ping-pong 特性端点。为保证带宽为常数，当 USB 器件接收当前数据有效负载时，微控制器必须读前一个主机发送的数据有效负载。因此要使用两个存储器段。一个给微控制器使用，另一个由 USB 器件锁定。

Figure 177. Ping-pong 端点数据出发送时的段交换

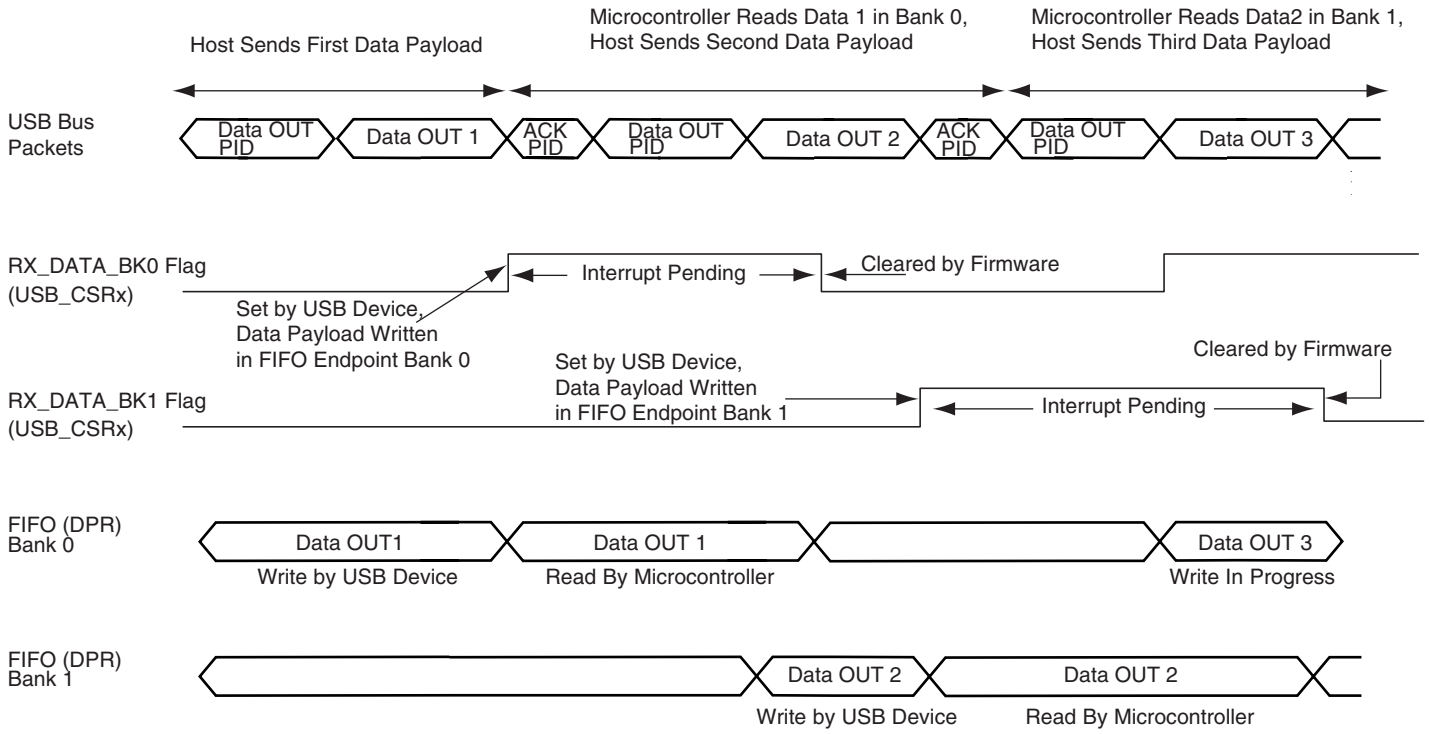


使用 ping-pong 端点时，数据出处理按以下步骤执行：

1. 主机产生数据出包。
2. USB 器件端点接收该包，写入端点 FIFO 段 0。
3. USB 器件向主机发送 ACK PID 包。主机可立即发送第二个数据出包。器件接收该包并复制到 FIFO 段 1。

4. 通过轮询端点USB_CSRx寄存器RX_DATA_BK0 ,通知微控制器USB器件已收到数据有效负载。 RX_DATA_BK0 置位时该端点中断挂起。
5. 读端点 USB_CSRx 寄存器的 RXBYTECNT 位可得到 FIFO 的可用字节数。
6. 微控制器将从端点存储器收到的数据发送打谱微控制器存储器。读端点 USB_FDRx 寄存器可得到接收到的数据。
7. 通过清零端点USB_CSRx寄存器的RX_DATA_BK0位 ,微控制器通知USB外设它已结束发送。
8. USB 外设器件可接收第三个数据出包，并复制到 FIFO 段 0。
9. 若已收到第二个数据出包 ,通过将端点USB_CSRx寄存器RX_DATA_BK1置位通知微控制器。当 RX_DATA_BK1 置位，端点中断挂起。
10. 微控制器将从端点存储器接收到的数据发送到微控制器存储器。读端点 USB_FDRx 寄存器可得到接收数据。
11. 通过清零端点USB_CSRx寄存器的RX_DATA_BK1位 ,微控制器通知USB器件它已完成发送。
12. USB 器件可接收第四个数据出包并复制到 FIFO 段 0。

Figure 178. Ping-pong 端点数据出发送



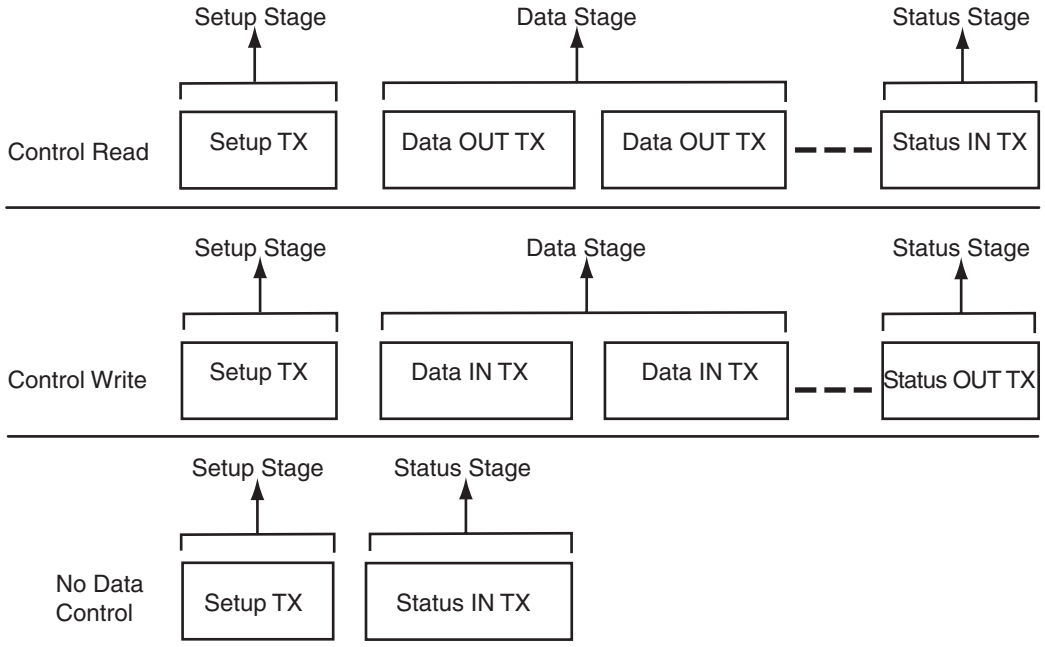
Note: 当 RX_DATA_BK0 或 RX_DATA_BK1 标志置位，中断挂起。

警告：当 RX_DATA_BK0 与 RX_DATA_BK1 均置位时，无法决定先对哪位清零。因此软件软件的内部计数器必须确保先清除 RX_DATA_BK0 ,然后再清除 RX_DATA_BK1。该状况可能在应用程序在别处忙且两段被 USB 主机填满时出现。一旦应用程序返回 USB 驱动器，两标志位置位。

状态处理

状态处理是控制发送中主机到器件处理的特殊类型。必须使用有 ping-pong 特性的端点执行控制发送。 USB 器件根据控制序列 (读或写) 发送或接收状态处理。

Figure 179. 控制读与写序列



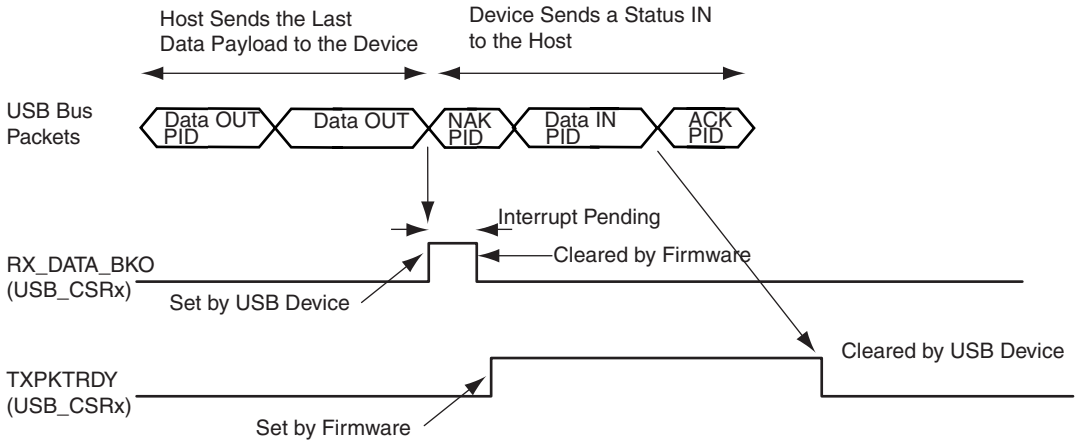
Notes: 1. 状态入阶段，主机使用DATA1 PID等待来自器件的0长度包(无数据的数据入处理)。详见*通用串行总线规范，Rev. 2.0*协议层。
2. 状态出状态，主机发射 0 长度包到器件 (无数据的数据出处理)。

状态入发送

一旦处理控制请求，器件向主机返回状态。这是 0 长度数据入处理

1. 微控制器等待端点USB_CSRx寄存器TXPKTRDY位清零(该步骤中，必须对TXPKTRDY清零，因为前一个处理是设置处理或数据出处理)。
2. 不需向 USB_FDRx 寄存器写入任何值，微控制器设置 TXPKTRDY。 USB 器件使用 DATA1 PID 产生数据入包。
3. 该包由主机应答，并设置 USB_CSRx 寄存器的 TXPKTRDY 位。

Figure 180. 数据输出后状态入处理

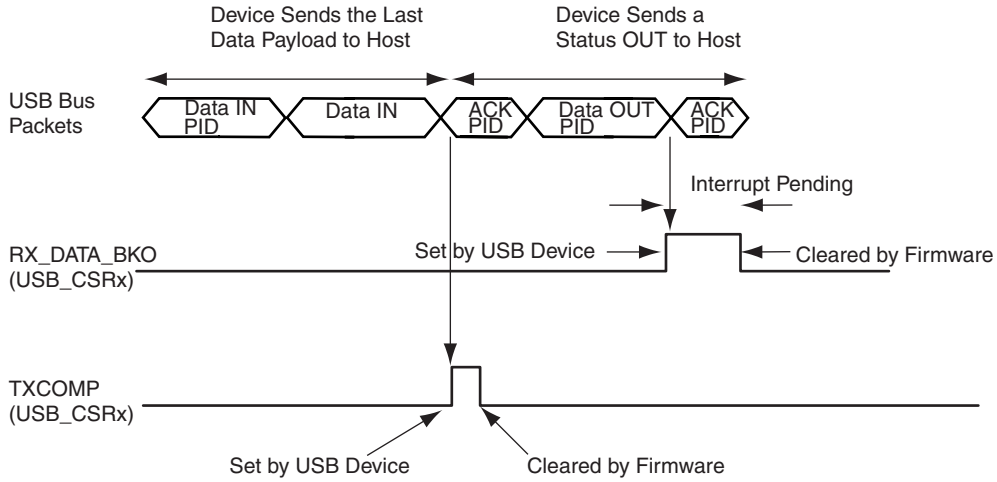


状态出发送

一旦处理控制请求且返回请求数据，主机通过发送 0 长度包应答。这是 0 长度数据出处理。

1. USB器件接收0长度包。它设置USB_CSRx寄存器中的RX_DATA_BK0标志并应答0长度包。
2. 通过轮询USB_CSRx寄存器的RX_DATA_BK0位,通知微控制器USB器件已接收到主机发送的 0 长度包。当 RX_DATA_BK0 置位时，中断挂起。端点 USB_BCR 寄存器收到的字节数为 0。
3. 微控制器必须对 RX_DATA_BK0 清零。

Figure 181. 数据入后状态出处理



停止握手

停止握手用于下述两种场合之一 (更多内容见*通用串行总线规范， Rev 2.0.*) :

- 当与端点相关的停止特性设置时使用功能停止 (更多内容见*通用串行总线规范， Rev 2.0*)。
- 中断当前请求，使用协议停止，仅适用于控制发送。

按以下步骤产生停止包：

1. 微控制器设置 USB_CSRx 寄存器的 FORCESTALL 标志。
2. 主机接收停止包。
3. 通过轮询 STALLSENT 设置来通知微控制器，器件已发送停止。当 STALLSENT 设置时端点中断挂起。微控制器必须对 STALLSENT 清零以清除中断。

当停止握手后收到设置处理时，必须对 STALLSENT 清零，以防止由于 STALLSENT 置位而引起的中断。

Figure 182. 停止握手 (数据入发送)

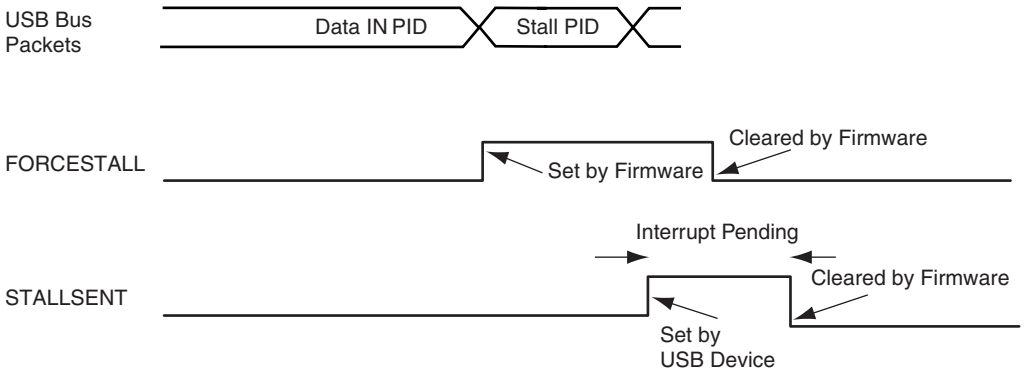
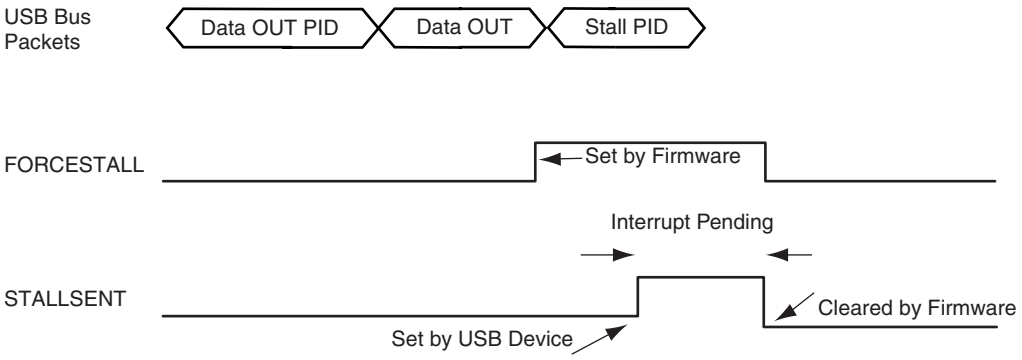


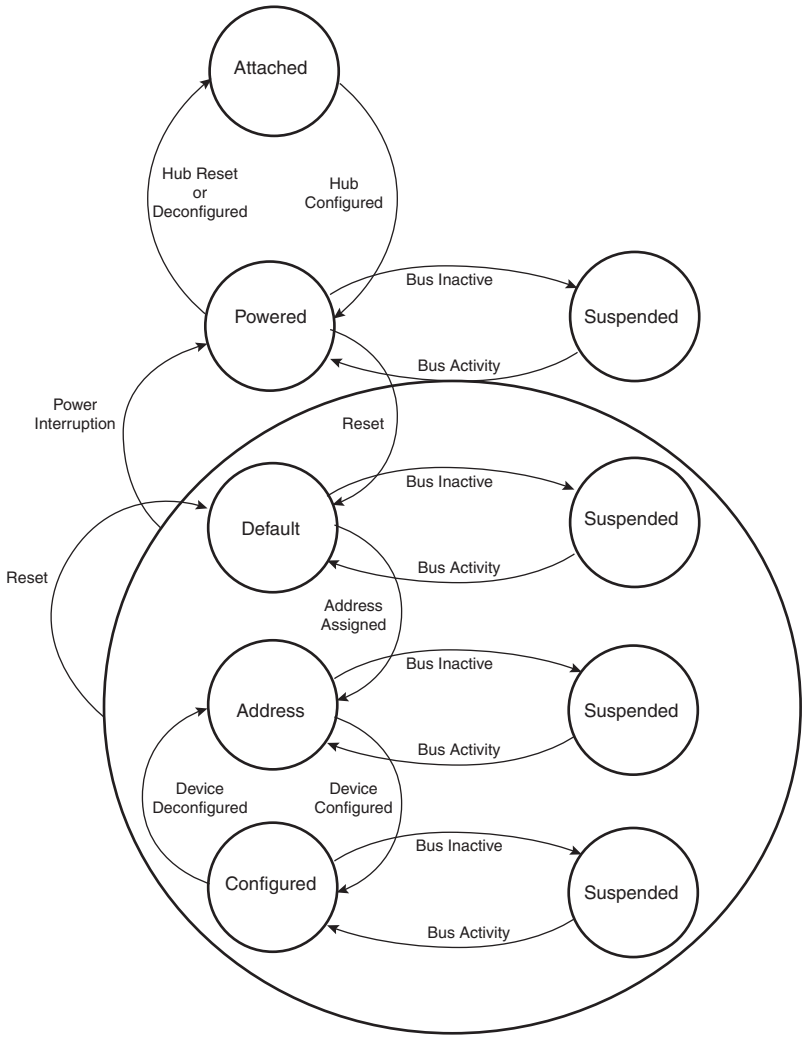
Figure 183. 停止握手 (数据出发送)



控制器件状态

USB 器件有几种可能状态，参见 *通用串行总线规范， Rev 2.0* 第九章。

Figure 184. USB 器件状态图



一个状态到另一个状态的转换取决于 USB 总线状态或通过默认端点 (端点 0) 的控制处理发送标准请求。

总线无效一周期后， UDP 器件进入挂起模式。强制接受 USB 主机挂起 / 恢复请求。挂起模式对总线供电应用要求非常严格；器件在 USB 总线上工作不超过 500 µA。

挂起模式时，主机通过发送恢复信号 (总线激活) 或 USB 器件向主机发送唤醒请求 以唤醒器件，例如通过移动 USB 鼠标来唤醒 PC。

唤醒特性并非对所有器件强制执行，并必须与主机协商。

由上电状态到默认状态

USB器件与USB主机连接后，它将等待一个总线结束复位。一旦在DP上检测到器件上拉，USB主机停止驱动复位状态。 UDP_ISR 中未屏蔽标志 ENDBURST 置位并触发中断。 UDP 软件使能默认端点，设置 UDP_CSR[0] 寄存器中 EPEDS 标志，并可通过在 UDP_IER 寄存器写 1 使能端点 0 中断 (可选)。由控制发送开始列举。

由默认状态到地址状态

在设置地址标准器件请求后，USB 主机进入地址状态。此前，它实现控制发送的状态入处理，即一旦收到 UDP_CSR[0] 寄存器的 TXCOMP 位并将其清零后，UDP 器件设置其新地址。

为进入地址状态，驱动器软件设置 UDP_GLB_STATE 寄存器的 FADDEN 标志，设置其新地址，并设置 UDP_FADDR 寄存器 FEN 位。

由地址状态到配置状态

一旦收到有效设置配置标准请求并应答，器件使能相应当前配置的端点。通过设置 UDP_CSRx 寄存器上的 EPEDS 及 EPTYPE 域实现，并可在 UDP_IER 寄存器中使能相应中断 (可选)。

挂起使能

当检测到挂起 (USB 总线上无总线活动) ，设置 UDP_ISR 寄存器上 RXSUSP 信号。若 UDP_IMR 寄存器相应位置位则触发中断。

通过写 UDP_ICR 寄存器清除该标志。然后器件进入挂起模式。例如，微控制器切换到慢时钟，禁用 PLL 及主振荡器，并进入空闲模式。它也可以切断板上其它器件。

USB 器件外设时钟也可被切断。但收发器及 USB 外设不能切断，否则无法检测恢复。

接收主机恢复

挂起模式下，USB 收发器及 USB 外设必须上电以检测 RESUME。但由于 WAKEUP 为异步信号，可能无法对 USB 外设计时。

一旦在总线上检测到恢复，设置 UDP_ISR 寄存器的 WAKEUP 信号。若 UDP_IMR 寄存器相应位置位，可能产生一个中断。该中断可用来唤醒内核，使能 PLL 及主振荡器并配置时钟。设置 UDP_ICR 寄存器 WAKEUP 后，WAKEUP 位必须尽快清零。

发送外部恢复

外部恢复由主机处理，通过设置 USB_GLB_STATE 寄存器 ESR 位来使能。外部 ext_resume_pin 上的异步事件产生 WAKEUP 中断。对于早期版本的 USP 外设，将立即在 USB 线上产生 K 状态。即 USB 器件必须可迅速对主机应答。新版本中，一旦与主机通信就绪，软件设置 UDP_GLB_STATE 寄存器的 RMWUPE 位。然后在总线上产生 K 状态。

通过设置 UDP_ICR 寄存器的 WAKEUP 位尽快将 WAKEUP 位清零。

USB 器件端口 (UDP) 用户接口

Table 83. USB 器件端口 (UDP) 寄存器映射

偏移	寄存器	名称	访问类型	复位状态
0x000	帧数寄存器	USB_FRM_NUM	读	0x0000_0000
0x004	全局状态寄存器	USB_GLB_STAT	读 / 写	0x0000_0010
0x008	功能地址寄存器	USB_FADDR	读 / 写	0x0000_0100
0x00C	保留	–	–	–
0x010	中断使能寄存器	USB_IER	写	
0x014	中断禁用寄存器	USB_IDR	写	
0x018	中断屏蔽寄存器	USB_IMR	读	0x0000_1200
0x01C	中断状态寄存器	USB_ISR	读	0x0000_0000
0x020	中断清除寄存器	USB_ICR	写	
0x024	保留	–	–	–
0x028	复位端点寄存器	USB_RST_EP	读 / 写	
0x02C	保留	–	–	–
0x030	端点 0 控制与状态寄存器	USB _CSR0	读 / 写	0x0000_0000
. . .	. . .			
See Note 1	端点 3 控制与状态寄存器	USB _CSR3	读 / 写	0x0000_0000
0x050	端点 0 FIFO 数据寄存器	USB_FDR0	读 / 写	0x0000_0000
. . .	. . .			
See Note 2	端点 3 FIFO 数据寄存器	USB_FDR3	读 / 写	0x0000_0000
0x070	保留	–	–	–
0x074	收发器控制寄存器	USB_TXVC	读 / 写	0x0000_0100
0x078 - 0x0FC	保留	–	–	–

Notes: 1. USB_CSRx 寄存器地址计算如下：0x030 + 4(端点序号 - 1)。
2. USB_FDRx 寄存器地址计算如下：0x050 + 4(端点序号 - 1)。

USB 帧数寄存器

寄存器名称 : USB_FRM_NUM

访问类型 : 只读

31	30	29	28	27	26	25	24
---	---	---	---	---	---	---	---

23	22	21	20	19	18	17	16
–	–	–	–	–	–	FRM_OK	FRM_ERR

15	14	13	12	11	10	9	8
–	–	–	–	–	FRM_NUM		

7	6	5	4	3	2	1	0
FRM_NUM							

- **FRM_NUM[10:0]: 包格式中定义的帧数目**

每有一帧，主机将这个 11 位值加一。每帧开始时更新此值。

在 SOF_EOP (帧开始，包结束) 时更新值。

- **FRM_ERR: 帧错误**

当接收到的 SOF 包中有错时，在 SOF_EOP 中的该位置位。

收到 SOF_PID 后该位复位。

- **FRM_OK: 帧正确**

当接收到的 SOF 包中无错时，在 SOF_EOP 中的该位置位。

收到 SOF_PID 后该位复位 (包标识)。

中断状态寄存器中，收到 SOF_PID 时更新 SOF 中断。该位置位不必等待 EOP。

Note: 在 8 位寄存器接口中， FRM_OK 为 FRM_NUM_H 的位 4， FRM_ERR 为 FRM_NUM_L 的位 3。

USB 全局状态寄存器

寄存器名称 : USB_GLB_STAT

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	RMWUPE	RSMINPR	ESR	CONFG	FADDEN

该寄存器用于按照 *USB 串行总线规范 , Rev.2.0* 第九章的说明来获得并设置器件。

- **FADDEN: 功能地址使能**

读：

0 = 器件不处于地址状态。

1 = 器件处于地址状态。

写：

0 = 无效，只有复位能将器件带回默认状态。

1 = 将器件设置为地址状态。在成功设置地址请求后出现。在此之前，必须用设置地址参数将 USB_FADDR 寄存器初始化。在设置 FADDEN 前设置地址必须完成状态阶段，详见*通用串行总线规范 , Rev.2.0* 第九章的说明。

- **CONFG: 配置**

读：

0 = 器件不处于配置状态。

1 = 器件处于配置状态。

写：

0 = 设置器件到非配置状态。

1 = 设置器件到配置状态。

当处于地址状态且接收到成功设置配置请求时将器件置于配置状态，详见*通用串行总线规范 , Rev.2.0* 第九章的说明。

- **ESR: 使能发送恢复**

0 = 禁用远程唤醒序列。

1 = 能进行远程唤醒并使能引脚发送恢复。

- **RSMINPR: 已向主机发送恢复**

读：

0 = 无效。

1 = 在远程唤醒中已向主机发送恢复。

- **RMWUPE: 远程唤醒使能**

0 = 收到任何主机包或 SOF 中断后必须清零。

1 = 若 ESR 使能，使能 USB 线上的 K 状态。

USB 功能地址寄存器

寄存器名称 : USB_FADDR

访问类型： 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	FEN

7	6	5	4	3	2	1	0
–	FADD						

- **FADD[6:0]: 功能地址值**

一旦从主机接收到设置地址请求，并处于无数据控制序列的状态阶段，固件必须对功能地址值编程，详见*通用串行总线规范, Rev.2.0* 的说明。上电或复位后，功能地址值设置为 0。

- **FEN: 功能使能**

读：

0 = 功能端点禁用。

1 = 功能端点使能。

写：

0 = 禁用功能端点。

1 = 默认值。

功能使能位 (FEN) 允许微控制器使能或禁用功能端点。从主机接收到复位后微控制器设置该位。一旦该位被设置，USB 器件被主机接受并可与主机传输数据包。

USB 中断使能寄存器

寄存器名称： USB_IER

访问类型： 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	WAKEUP	–	SOFINT	EXTRSM	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
–	–			EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: 使能端点 0 中断**
- **EP1INT: 使能端点 1 中断**
- **EP2INT: 使能端点 2 中断**
- **EP3INT: 使能端点 3 中断**

0 = 无效。

1 = 使能相应端点中断。

- **RXSUSP: 使能 USB 挂起中断**

0 = 无效。

1 = 使能 USB 挂起中断。

- **RXRSM: 使能 USB 恢复中断**

0 = 无效。

1 = 使能 USB 恢复中断。

- **EXTRSM: 使能外部恢复中断**

0 = 无效。

1 = 使能外部恢复中断。

- **SOFINT: 使能帧起始中断**

0 = 无效。

1 = 使能帧起始中断。

- **WAKEUP: 使能 USB 总线唤醒中断**

0 = 无效。

1 = 使能 USB 总线中断。

USB 中断禁用寄存器

寄存器名称 : USB_IDR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	WAKEUP	–	SOFINT	EXTRSM	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
–	–			EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: 禁用端点 0 中断**
- **EP1INT: 禁用端点 1 中断**
- **EP2INT: 禁用端点 2 中断**
- **EP3INT: 禁用端点 3 中断**

0 = 无效。

1 = 禁用相应端点中断。

- **RXSUSP: 禁用 USB 挂起中断**

0 = 无效。

1 = 禁用 USB 挂起中断。

- **RXRSM: 禁用 USB 恢复中断**

0 = 无效。

1 = 禁用 USB 恢复中断。

- **EXTRSM: 禁用外部恢复中断**

0 = 无效。

1 = 禁用外部恢复中断。

- **SOFINT: 禁用帧起始中断**

0 = 无效。

1 = 禁用帧起始中断。

- **WAKEUP: 禁用 USB 总线唤醒中断**

0 = 无效。

1 = 禁用 USB 总线中断。

USB 中断屏蔽寄存器

寄存器名称： USB_IMR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	WAKEUP	–	SOFINT	EXTRSM	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
–	–			EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: 屏蔽端点 0 中断**
- **EP1INT: 屏蔽端点 1 中断**
- **EP2INT: 屏蔽端点 2 中断**
- **EP3INT: 屏蔽端点 3 中断**

0 = 禁用相应端点中断。

1 = 使能相应端点中断。

- **RXSUSP: 屏蔽 USB 挂起中断**

0 = 禁用 USB 挂起中断。

1 = 使能 USB 挂起中断。

- **RXRSM: 屏蔽 USB 恢复中断**

0 = 禁用 USB 恢复中断。

1 = 使能 USB 恢复中断。

- **EXTRSM: 屏蔽外部恢复中断**

0 = 禁用外部恢复中断。

1 = 使能外部恢复中断。

- **SOFINT: 屏蔽帧起始中断**

0 = 禁用帧起始中断。

1 = 使能帧起始中断。

- **WAKEUP: USB 总线唤醒中断**

0 = 禁用 USB 总线唤醒中断。

1 = 使能 USB 总线唤醒中断。

Note: 当 USB 块处于挂起模式时，应用程序可能关闭 USB 逻辑。此时，所有 USB 主机恢复请求必须记录，因此，USB_IMR 寄存器的 RXRSM 位复位值必须使能。

USB 中断状态寄存器

寄存器名称： USB_ISR

访问类型： 只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	WAKEUP	ENDBUSRES	SOFINT	EXTRSM	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
–	–			EP3INT	EP2INT	EP1INT	EP0INT

- **EP0INT: 中断 0 中断状态**

0 = 无端点 0 中断挂起。

1 = 端点 0 中断已发生。

几个信号可产生该中断，通过读 USB_CSR0 可得：

RXSETUP 置为 1

RX_DATA_BK0 置为 1

RX_DATA_BK1 置为 1

TXCOMP 置为 1

STALLSENT 置为 1

EP0INT 为 粘着位。写相应 USB_CSR0 位对 EP0INT 清零前中断有效。

- **EP1INT: 端点 1 中断状态**

0 = 无端点 1 中断挂起。

1 = 端点 1 中断已发生。

几个信号可产生该中断，通过读 USB_CSR1 可得：

RXSETUP 置为 1

RX_DATA_BK0 置为 1

RX_DATA_BK1 置为 1

TXCOMP 置为 1

STALLSENT 置为 1

EP1INT 为 粘着位。写相应 USB_CSR1 位对 EP1INT 清零前中断有效。

- **EP2INT: 端点 2 中断状态**

0 = 无端点 2 中断挂起。

1 = 端点 2 中断已发生。

几个信号可产生该中断，通过读 USB_CSR2 可得：

RXSETUP 置为 1

RX_DATA_BK0 置为 1

RX_DATA_BK1 置为 1

TXCOMP 置为 1

STALLSENT 置为 1

EP2INT 为 粘着位。写相应 USB_CSR2 位对 EP2INT 清零前中断有效。

- **EP3INT: 端点 3 中断状态**

0 = 无端点 3 中断挂起。

1 = 端点 3 中断已发生。

几个信号可产生该中断，通过读 USB_CSR3 可得：

RXSETUP 置为 1

RX_DATA_BK0 置为 1

RX_DATA_BK1 置为 1

TXCOMP 置为 1

STALLSENT 置为 1

EP3INT 为 粘着位。写相应 USB_CSR3 位对 EP3INT 清零前中断有效。

- **RXSUSP: USB 挂起中断状态**

0 = 无 USB 挂起中断等待。

1 = USB 挂起中断已发生。

当 USB 器件检测到 3ms 无工作时设置该位。 USB 器件进入挂起模式。

- **RXRSM: USB 恢复中断状态**

0 = 无 USB 恢复中断等待。

1 =USB 恢复中断已发生。

当 USB 器件端口检测到 USB 恢复信号时设置该位。

- **EXTRSM: 外部恢复中断状态**

0 = 无外部恢复中断等待。

1 = 外部恢复中断已发生。

挂起模式下，在 send_resume 检测到一个异步上升沿时发生中断。

若 RMWUPE = 1， USB 总线上发送一个恢复状态。

- **SOFINT: 帧开始中断状态**

0 = 无帧开始中断等待。

1 = 帧开始中断已发生。

每次检测到 SOF 时发生中断。它可作为同步信号使用同步端点。

- **ENDBUSRES: 总线结束复位中断状态**

0 = 无总线结束复位中断等待。

1 = 总线结束复位中断已发生。

USB 复位序列结束后发生中断。 USB 器件必须在端点 0 准备接收请求。主机开始列举，然后开始进行配置。

- **WAKEUP: USB 恢复中断状态**

0 = 无唤醒中断等待。

1 = 上次清零后出现唤醒中断 (USB 主机发送 RESUME 或 RESET)。

USB 中断清除寄存器

寄存器名称 : USB_ICR

访问类型 : 只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	WAKEUP	ENDBURST	SOFINT	EXTRSM	RXRSM	RXSUSP

7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **RXSUSP: 清除 USB 挂起中断**

0 = 无效。

1 = 清除 USB 挂起中断。

- **RXRSM: 清除 USB 恢复中断**

0 = 无效。

1 = 清除 USB 恢复中断。

- **EXTRSM: 清除外部恢复中断**

0 = 无效。

1 = 清除外部恢复中断。

- **SOFINT: 清除帧开始中断**

0 = 无效。

1 = 清除帧开始中断。

- **ENDBURST: 清除总线复位结束中断**

0 = 无效。

1 = 清除总线复位结束中断。

- **WAKEUP: 清除唤醒中断**

0 = 无效。

1 = 清除唤醒中断。

USB 复位端点寄存器

寄存器名称 : USB_RST_EP

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
–	–			EP3	EP2	EP1	EP0

- **EP0: 复位端点 0**
- **EP1: 复位端点 1**
- **EP2: 复位端点 2**
- **EP3: 复位端点 3**

该标志用于复位与端点相关的 FIFO 及 UDP_CSRx 寄存器中的 RXBYTECOUNT 位。它还复位数据切换到 DATA0。它在删除 BULK 端点 HALT 状态后有用。参见 *USB 串行总线规范， Rev.2.0* 的 5.8.5 节。

警告：复位结束时该标志必须清除。不清除 USB_CSRx 标志。

0 = 无复位。

1 = 强制将相应端点 FIF0 指向 0，因此 USB_CSRx 寄存器 RXBYTECNT 域为 0。

USB 端点控制与状态寄存器

寄存器名称 : USB_CSRx [x = 0..3]

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	RXBYTECNT		

23	22	21	20	19	18	17	16
RXBYTECNT							

15	14	13	12	11	10	9	8
EPEDS	–	–	–	DTGLE	EPTYPE		

7	6	5	4	3	2	1	0
DIR	RX_DATA_ BK1	FORCE STALL	TXPKTRDY	STALLSENT ISOERROR	RXSETUP	RX_DATA_ BK0	TXCOMP

- **TXCOMP: 产生一个有前数据写入 DPR 的入包**

该标志设置为 1 时产生中断。

写 (由固件清零) ;

0 = 清除标志，清除中断。

1 = 无效。

读 (由 USB 外设设置) :

0 = 主机未应答数据入处理。

1 = 数据入处理完成，主机应答。

数据入处理设置 TXPKTRDY 后，器件固件等待 TXCOMP 以确保主机已应答处理。

- **RX_DATA_BK0: 接收数据段 0**

该标志设置为 1 时产生中断。

写 (由固件清零) ;

0 = 通知 USB 外设器件， FIFO 段 0 数据已被读取。

1 = 无效。

读 (由 USB 外设设置) :

0 = FIFO 段 0 未收到数据包。

1 = 已收到数据包，并已存入 FIFO 段 0。

当器件固件已轮询该位或由该信号中断，必须将数据由 FIFO 送到微控制器存储器。 RXBYTCENT 域中收到的字节数有效。通过 USB_FDRx 寄存器读 FIFO 段 0 值。一旦发送完成 ,器件固件必须通过清除 RX_DATA_BK0 将 USB 外设器件段 0 释放。

- **RXSETUP: 向主机发送 STALL(控制端点)**

该标志设置为 1 时产生中断。

读 :

0 = 无有效设置包。

1 = 主机已发送设置包并在 FIFO 中有效。

写 :

0 = 器件固件通知 USB 外设器件，它已读取 FIFO 中设置数据。

1 = 无效。

该标志用来通知 USB 器件固件，主机已收到有效设置数据包并由 USB 器件成功接收。 USB 器件固件可通过读 USB_FDRx 寄存器将 FIFO 中设置数据发送到微控制器处理器。一旦发送完成， RXSETUP 必须由器件固件清除。

当 RXSETUP 置位时，不接收随后的数据出数据。

- **STALLSENT: 发送停止 (控制、批中断端点) / ISOERROR (同步端点)**

该标志设置为 1 时产生中断。

STALLSENT：结束停止握手。

读：

0 = 主机未应答 STALL。

1 = 主机应答 STALL。

写：

0 = 复位 STALLSENT 标志，清除中断。

1 = 无效。

强制器件固件清除该标志，否则中断保持。

关于 STALL 握手，参见*通用串行总线规范， Rev. 2.0* 8.4.5 及 9.4.5 节。

ISOERROR：同步发送中检测到 CRC 错误。

读：

0 = 之前的同步传输中无错误。

1 = 检测到 CRC 错误，FIFO 中可用数据被破坏。

写：

0 = ISOERROR 标志复位，清除中断。

1 = 无效。

- **TXPKTRDY: 发送包就绪**

该标志由 USB 器件清除。

该标志由 USB 器件固件置位。

读：

0 = 数据值可写入 FIFO。

1 = 数据值不能写入 FIFO。

写：

0 = 无效。

1 = 新数据有效负载通过固件写入 FIFO 并即将发送。

该标志用来产生数据入处理 (器件到主机)。器件固件检查它能否在 FIFO 中写入数据有效负载，检查 TXPKTRDY 是否清零。通过写 USB_FDRx 寄存器实现数据到 FIFO 的发送。一旦数据有效否则发送到 FIFO，固件通知 USB 器件将 TXPKTRDY 设置为 1 。 USB 总线处理开始。一旦主机收到数据有效负载， TXCOMP 置位。

- **FORCESTALL: 强制停止 (用于控制、批及同步端点)**

只写

0 = 无效。

1 = 向主机发送 STALL。

关于 STALL 握手，参见*通用串行总线规范， Rev. 2.0* 8.4.5 及 9.4.5 节。

控制端点：数据筹备与状态筹备过程中，表明微控制器不能完成请求。

批与中断端点：通知主机端点已停止。

主机应答 STALL， STALLSENT 标志通知器件固件。

- **RX_DATA_BK1: 接收数据段 1 (只用于有 ping-pong 特性的端点)**

该标志设置为 1 时产生中断。

写 (由固件清零) ；

0 = 通知 USB 器件，FIFO 段 1 数据已被读取。

1 = 无效。

读 (由 USB 外设设置) :

0 = FIFO 段 1 未收到数据包。

1 = 已收到数据包，并已存入 FIFO 段 1。

当器件固件已轮询该位或由该信号中断，必须将数据由 FIFO 送到微控制器存储器。 RXBYTCENT 域中收到的字节数有效。通过 USB_FDRx 寄存器读 FIFO 段 1 值。一旦发送完成 ,器件固件必须通过清除 RX_DATA_BK1 将 USB 外设器件段 1 释放。

- **DIR: 发送方向 (仅对控制端点有效)**

读 / 写

0 = 在控制数据筹备时允许数据出处理。

1 = 控制数据筹备时使能数据入处理。

关于控制数据筹备，参见*通用串行总线规范， Rev. 2.0* 8.5.3 节。

在设置筹备结束清除 USB_CSRx/RXSETUP 前该位必须置位。根据设置数据包发送请求，数据筹备为器件到主机 (DIR = 1) 或主机到器件 (DIR = 0) 数据发送。状态筹备时不需检验该位反向。

- **EPTYPE[2:0]: 端点类型**

读 / 写

000	控制
001	同步出
101	同步入
010	批出
110	批入
011	中断出
111	中断入

- **DTGLE: 数据切换**

只读

0 = 识别 DATA0 包。

1 = 识别 DATA1 包。

更多 DATA0、 DATA1 包定义见*通用串行总线规范， Rev. 2.0* 第八章。

- **EPEDS: 端点使能禁用**

读：

0 = 端点禁用。

1 = 端点使能。

写：

0 = 禁用端点。

1 = 使能端点。

- **RXBYTECNT[10:0]: FIFO 中可用字节数**

只读

当主机向器件发送数据包时， USB 器件将数据存入 FIFO 并通知微控制器。微控制器可通过读取 USB_FDRx 寄存器的 RXBYTECENT 字节载入 FIFO 中数据。

USB FIFO 数据寄存器

寄存器名称 : USB_FDRx [x = 0..3]

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–

23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–

15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–

7	6	5	4	3	2	1	0
FIFO_DATA							

- **FIFO_DATA[7:0]: FIFO 数据值**

微控制器可通过该寄存器将数据推入或弹出 FIFO。

相应 USB_CSRx 寄存器的 RXBYTECNT 中是由 FIFO 中读取的字节数 (主机发送)。

能写入的最大字节数由标准端点描述符的最大包尺寸确定。它不能大于相关端点物理存储器大小，详见*通用串行总线规范 , Rev. 2.0*。

USB 收发控制寄存器

寄存器名称 : USB_TXVC

访问类型 : 读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	TXVDIS
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–

- **TXVDIS: 收发器禁用**

当 UDP 禁用，通过禁用内置收发器，可显著降低功耗。通过设置 TXVDIS 域实现。

为使能收发器，必须将 TXVDIS 清零。

模数转换器 (ADC)

概述

ADC是基于连续近似寄存器(SAR)的10位模数转换器(ADC)。它还集成了一个8到1的模拟多路复用器，可实现八条模拟线的模数转换。转换由 0V 到 ADVREF。

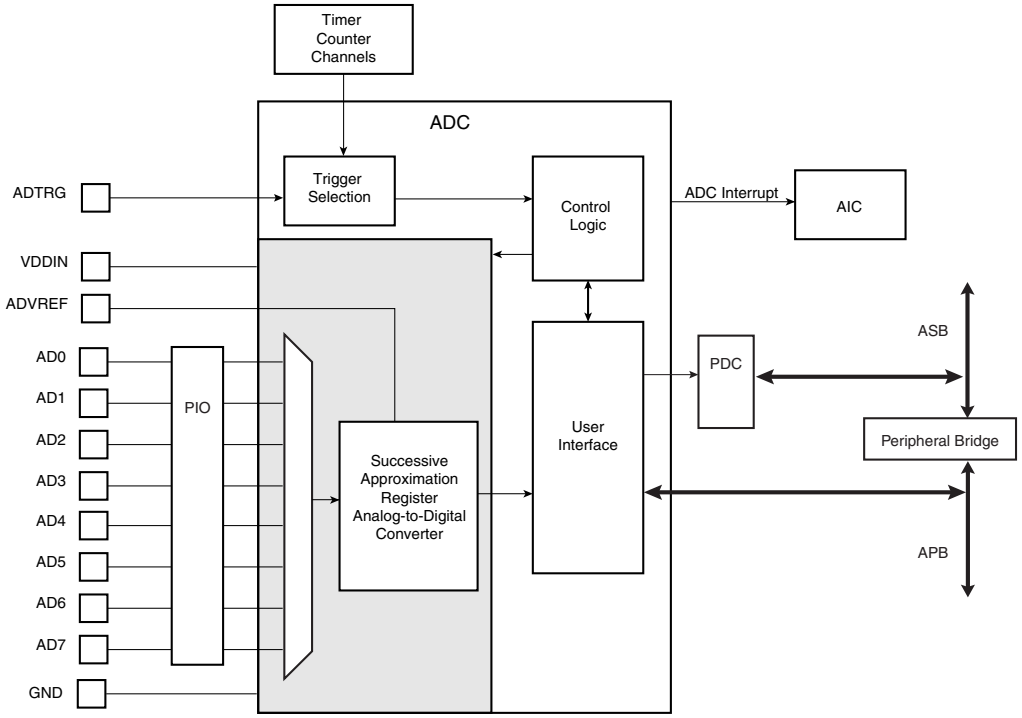
ADC支持8位或10位分辨率模式，并且转换结果进入一个所有通道可用的通用寄存器中，即通道专用寄存器。可配置为软件触发、外部触发 ADTRG 引脚上升沿或内部触发定时计数器输出。

ADC 还集成休眠模式与转换序列发生器，并与 PDC 通道连接。这些特性可减低功耗及处理器干涉。

最后，用户可配置 ADC 时间，如启动时间及采样与保持时间。

方框图

Figure 185. 模数转换器框图



信号说明

Table 84. ADC 引脚说明

引脚名称	说明
VDDIN	模拟电源
ADVREF	参考电压
AD0 - AD7	模拟输入通道
ADTRG	外部触发器

附属产品

电源管理

普通模式下，首次转换后 ADC 自动定时。休眠模式下，每次转换后 ADC 时钟自动停止。由于逻辑小且 ADC 单元可进入休眠模式，电源管理控制器对 ADC 工作无效。

中断源

ADC 中断线与高级中断控制器的一条内部源连接。使用 ADC 中断请求前须先对 AIC 编程。

模拟输入

引脚 AD0 到 AD7 可与 PIO 线复用。此时，通过写 ADC_CHER 寄存器使能相应通道时，自动完成对 ADC 输入分配。默认情况下，复位后， PIO 线配置为输入且其上拉使能，而 ADC 输入与 GND 连接。

I/O 线

通过 PIO 控制器设置，引脚 ADTRG 可能与其它外设功能共用。此时，应设置 PIO 控制器，将 ADTRG 引脚分配为 ADC 功能。

定时器触发

根据用户需要决定定时器计数器是否作为硬件触发器。因此，某些或所有定时器计数器可不连接。

转换执行

关于 ADC 执行及电气特性，见 “ADC Characteristics” on page 453。

功能说明

模数转换

ADC 使用 ADC 时钟执行转换。将单模拟信号转换为一个 10 位数字数据需要采样与保持时钟周期，定义见 “ADC 模式寄存器 ” on page 434 SHTIM 域，及 10 个 ADC 时钟周期。ADC 时钟频率由模式寄存器 (ADC_MR) 的 PRESCAL 域选定。

若 PRESCAL 为 0 ,ADC 时钟为 MCK/2 ；若 PRESCAL 为 63 (0x3F) ,ADC 时钟为 MCK/128。PRESCAL 必须根据产品定义部分参数编程，以提供 ADC 时钟频率。

转换参考

转换在 0V 到 ADVREF 引脚间执行。模拟输入在这些电压中的转换是线性转换。

转换分辨率

ADC支持8位或10位的分辨率。通过设置ADC模式寄存器(ADC_MR)的LOWRES位执行对8位的选择。默认情况下，复位后，分辨率最高，且数据寄存器中的 DATA 域完全使用。通过设置 LOWRES 位， ADC 切换到最低分辨率，且转换结果可从数据寄存器的低八位中读出。对应于 ADC_CDR 寄存器的 DATA 域最高两位及 ADC_LCDR 寄存器的 LDATA 位为 0。

此外，当 PDC 通道与 ADC 连接， 10 位分辨率设置发送请求大小到 16 位。设置 LOWRES 位自动切换到 8 位数据发送。此时，优化目的缓冲器。

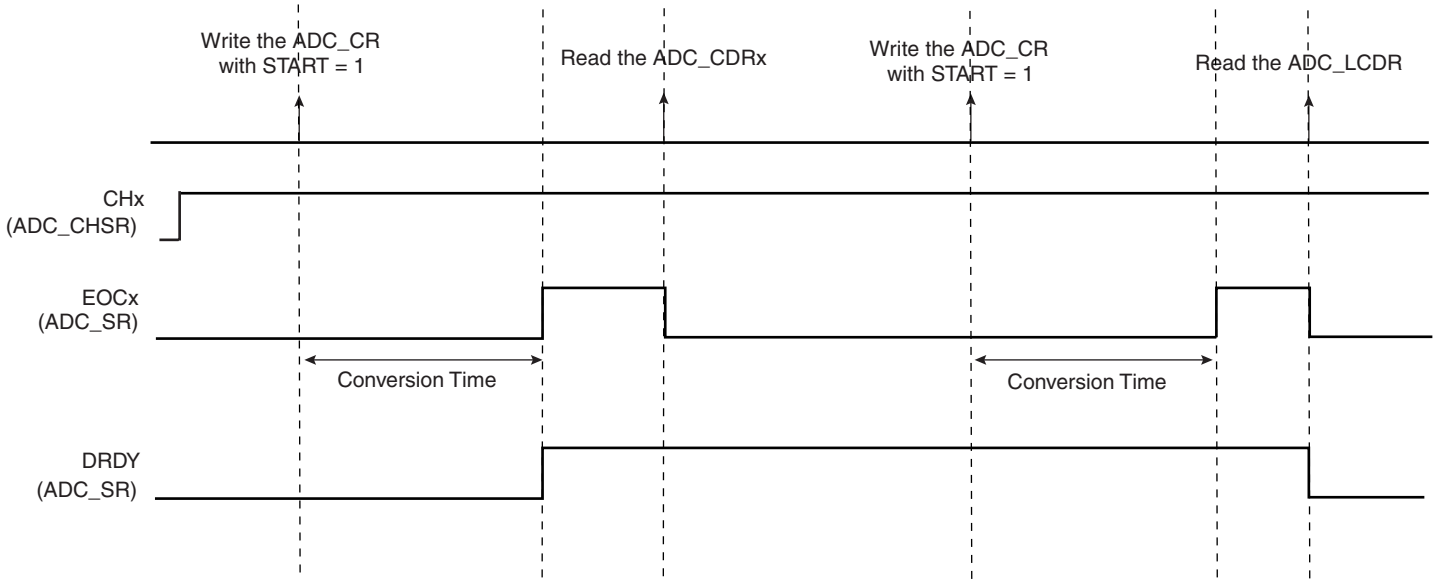
转换结果

当转换完成， 10 位数字结果存于当前通道的通道数据寄存器 (ADC_CDR) 及 ADC 最后转换数据寄存器 (ADC_LCDR) 中。

设置状态寄存器 (ADC_SR) 通道 EOC 位及 DRDY 位。与 PDC 通道连接时， DRDY 上升触发数据发送请求。任何情况下， EOC 与 DRDY 均可触发中断。

读取任意一个 ADC_CDR 寄存器将清除相应的 EOC 位。读取 ADC_LCDR 清除 DRDY 位及对应最后转换通道的 EOC 位。

Figure 186. EOCx 与 DRDY 标志动作

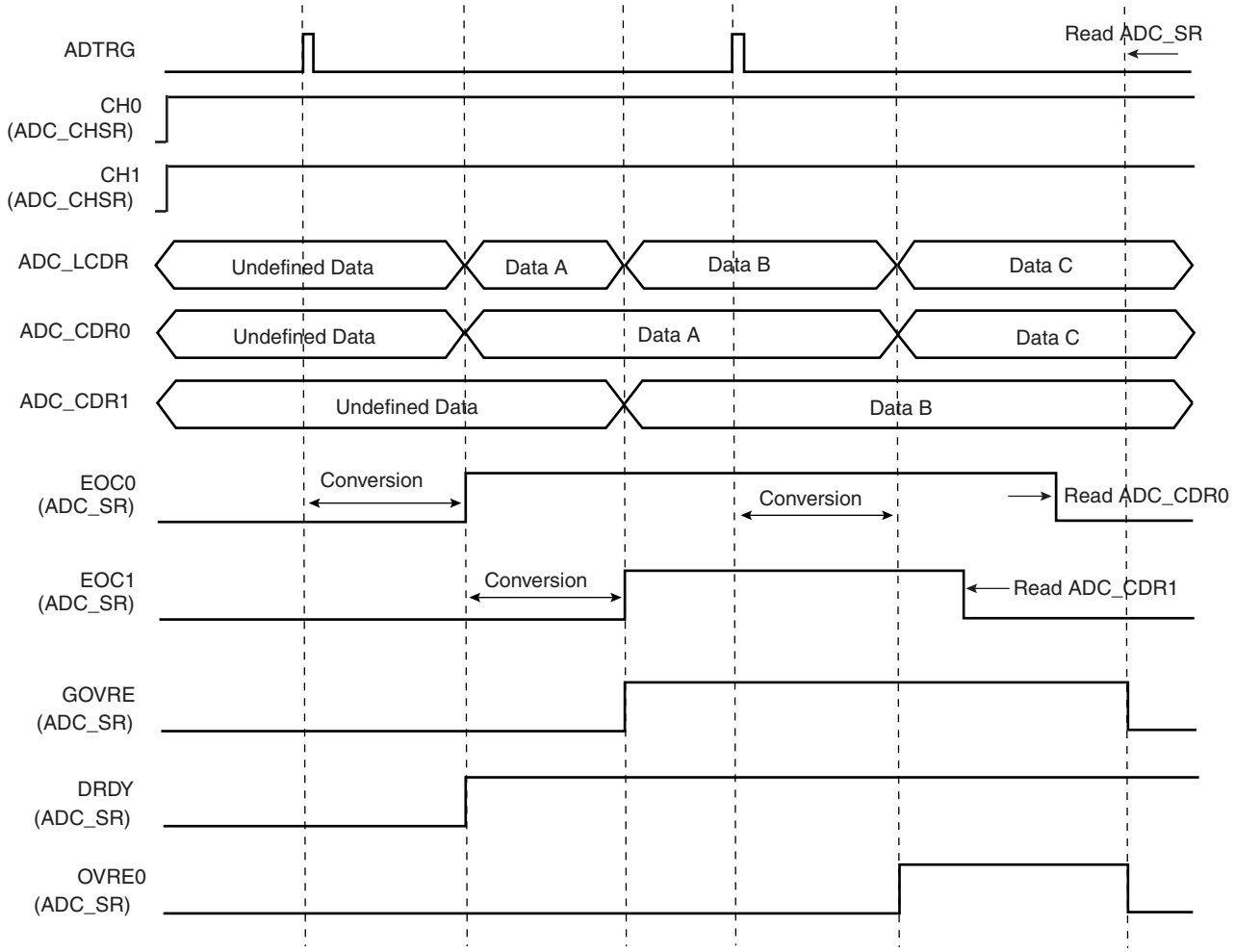


若在更多输入数据转换前未读取 ADC_CDR , 状态寄存器 (ADC_SR) 中相应的溢出错误 (OVRE) 标志置位。

当 DRDY 为高时转换新数据，使用相同方法设置 ADC_SR 寄存器中的 GOVRE(通用溢出错误) 位。

当读取 ADC_SR 时， OVRE 与 GOVRE 标志自动清零。

Figure 187. GOVRE 与 OVREx 标志动作



警告 : 若转换时相应通道禁用或转换时先禁用然后重新使能 , 它的相关数据及 ADC_SR 中 EOC 与 OVRE 标志结果未知。

转换触发器

使用软件或硬件触发器启动模拟通道转换。在控制寄存器 (ADC_CR) 的 START 位写 1 可实现软件触发。

硬件触发器是定时器计数器通道 TIOA 输出之一 , 或 ADC 外部触发输入 (ADTRG)。硬件触发器的选择由 ADC_MR 寄存器的 TRGSEL 域实现。选定的硬件触发器通过 ADC_MR 寄存器的 TRGEN 位使能。

若选择硬件触发器 , 在选定信号的每个上升沿进行转换。若选择一个 TIOA 输出 , 相应的定时器计数器必须编程为波形模式。

对于所有通道只需要一个启动命令来初始化转换序列。 ADC 硬件逻辑对工作通道自动执行转换，然后等待新请求。通道使能 (ADC_CHER) 与通道禁用 (ADC_CHDR) 寄存器使能模拟通道的独立使能或禁用。

若 ADC 在 PDC 上使用，只执行使能通道转换数据的发送且结果数据缓冲器据此说明。

警告：使能硬件触发器不会禁用软件触发功能。因此，若选择硬件触发，转换启动可由硬件或软件触发器来初始化。

休眠模式与转换序列发生器

ADC 休眠模式下最省电的部分是当转换未使用 ADC 时使其无效。通过设置 ADC_MR 寄存器的 SLEEP 位选择休眠模式。

SLEEP 模式由转换发生器自动管理，它可在最低功耗下自动处理转换。

当开始转换请求出现， ADC 自动激活。由于模拟单元需要一个启动时间，所以在此时间内逻辑等待并启动使能通道的转换。当所有转换完成， ADC 在新的触发到来前无效。序列中出现的触发不记。

转换序列发生器允许使用最少的处理器干涉及最优化的功耗条件下自动处理。转换序列可使用定时器 / 计数器输出周期性执行。通过 PDC 可自动处理对几个采样值的周期性采样，而不需使用处理器。

Note: 普通模式与休眠模式下，参考电压引脚始终保持连接。

ADC 时间

每个 ADC 有其自身最小启动时间，通过 ADC_MR 的 STARTUP 域编程设定。

同样，ADC 需要最小采样与保持时间以确保在两通道间选择最好的转换值。该时间在 ADC_MR 的 SHTIM 域编程设定。

警告：ADC 中没有隔离源的输入缓冲放大器。因此必须在 SHTIM 域中写入精确值，见产品手册 DC 特性部分。

模数转换器 (ADC) 用户接口

Table 85. ADC 寄存器映射

偏移	寄存器	名称	访问类型	复位状态
0x00	控制寄存器	ADC_CR	只写	–
0x04	模式寄存器	ADC_MR	读 / 写	0x00000000
0x08	保留	–	–	–
0x0C	保留	–	–	–
0x10	通道使能寄存器	ADC_CHER	只写	–
0x14	通道禁用寄存器	ADC_CHDR	只写	–
0x18	通道状态寄存器	ADC_CHSR	只读	0x00000000
0x1C	状态寄存器	ADC_SR	只读	0x000C0000
0x20	最后转换数据寄存器	ADC_LCDR	只读	0x00000000
0x24	中断使能寄存器	ADC_IER	只写	–
0x28	中断禁用寄存器	ADC_IDR	只写	–
0x2C	中断屏蔽寄存器	ADC_IMR	只读	0x00000000
0x30	通道数据寄存器 0	ADC_CDR0	只读	0x00000000
0x34	通道数据寄存器 1	ADC_CDR1	只读	0x00000000
0x38	通道数据寄存器 2	ADC_CDR2	只读	0x00000000
0x3C	通道数据寄存器 3	ADC_CDR3	只读	0x00000000
0x40	通道数据寄存器 4	ADC_CDR4	只读	0x00000000
0x44	通道数据寄存器 5	ADC_CDR5	只读	0x00000000
0x48	通道数据寄存器 6	ADC_CDR6	只读	0x00000000
0x4C	通道数据寄存器 7	ADC_CDR7	只读	0x00000000
0x50 - 0xFC	保留	–	–	–

ADC 控制寄存器

寄存器名称 :ADC_CR

访问类型 :只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
–	–	–	–	–	–	START	SWRST

- **SWRST: 软件复位**

0 = 无效。

1 = 复位 ADC 模拟硬件复位。

- **START: 开始转换**

0 = 无效。

1 = 开始模数转换。

ADC 模式寄存器

寄存器名称 :ADC_MR

访问类型 :读 / 写

31	30	29	28	27	26	25	24
–	–	–	–	SHTIM			
23	22	21	20	19	18	17	16
–	–	–	STARTUP				
15	14	13	12	11	10	9	8
–	–	PRESCAL					
7	6	5	4	3	2	1	0
–	–	SLEEP	LOWRES	TRGSEL			TRGEN

- **TRGEN: 触发使能**

TRGEN	选择 TRGEN
0	硬件触发禁用。只能通过软件启动转换。
1	通过 TRGSEL 域选择硬件触发使能。

- **TRGSEL: 触发选择**

TRGSEL			选择 TRGSEL
0	0	0	TIOA Ouput of the Timer Counter Channel 0
0	0	1	TIOA Ouput of the Timer Counter Channel 1
0	1	0	TIOA Ouput of the Timer Counter Channel 2
0	1	1	保留
1	0	0	保留
1	0	1	保留
1	1	0	外部触发器
1	1	1	保留

- **LOWRES: 分辨率**

LOWRES	选择分辨率
0	10 位分辨率
1	8 位分辨率

- **SLEEP: 休眠模式**

SLEEP	选择模式
0	普通模式
1	休眠模式

- **PRESCAL: 预分频器速率选择**

ADC 时钟 = MCK / ((PRESCAL+1) * 2)

- **STARTUP: 启动时间**

启动时间 = (STARTUP+1) * 8 / ADC 时钟

- **SHTIM: 采样与保持时间**

采样与保持时间 = (SHTIM+1) / ADC 时钟

ADC 通道使能寄存器

寄存器名称 :ADC_CHER

访问类型 :只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHx: 通道 x 使能**

0 = 无效。

1 = 使能相应通道。

ADC 通道禁用寄存器

寄存器名称 :ADC_CHDR

访问类型 :只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHx: 通道 x 禁用**

0 = 无效。

1 = 禁用相应通道。

警告 : 若转换时相应通道禁用或转换时先禁用然后重新使能，它的相关数据及 ADC_SR 中 EOC 与 OVRE 标志结果未知。

ADC 通道状态寄存器

寄存器名称 :ADC_CHSR

访问类型 :只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

- **CHx: 通道 x 状态**

0 = 相应通道禁用。

1 = 相应通道使能。

ADC 状态寄存器

寄存器名称 :ADC_SR

访问类型 :只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	RXBUFF	ENDRX	GOVRE	DRDY
15	14	13	12	11	10	9	8
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: x 转换结束**

0 = 相应模拟通道禁用或转换未结束。

1 = 相应模拟通道使能或转换完成。

- **OVREx: x 溢出错误**

0 = 上次读 ADC_SR 后，相应通道无溢出错误。

1 = 上次读 ADC_SR 后，相应通道有溢出错误。

- **DRDY: 数据就绪**

0 = 上次读 ADC_LCDR 后，无数据转换。

1 = 上次读 ADC_LCDR 后，至少有一个数据转换。

- **GOVRE: 通用溢出错误**

0 = 上次读 ADC_SR 后，无溢出错误。

1 = 上次读 ADC_SR 后，至少有一个溢出错误。

- **ENDRX: RX 缓冲器结束**

0 = 上次写 ADC_RCR 或 ADC_RNCR 后，接收计数寄存器未达到 0。

1 = 上次写 ADC_RCR 或 ADC_RNCR 后，接收计数寄存器已达到 0。

- **RXBUFF: RX 缓冲器满**

0 = ADC_RCR 或 ADC_RNCR 值不为 0。

1 = ADC_RCR 与 ADC_RNCR 值均为 0。

ADC 最后转换数据寄存器

寄存器名称 :ADC_LCDR

访问类型 :只读

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	–	–	–	LDATA	
7	6	5	4	3	2	1	0
LDATA							

- **LDATA: 最后数据转换**

转换结束后将模数转换数据置于该寄存器并保持到新转换结束。

ADC 中断使能寄存器

寄存器名称 :ADC_IER

访问类型 :只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	RXBUFF	ENDRX	GOVRE	DRDY
15	14	13	12	11	10	9	8
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: x 转换结束中断使能**
- **OVREx: x 溢出错误中断使能**
- **DRDY: 数据就绪中断使能**
- **GOVRE: 通用溢出错误中断使能**
- **ENDRX: 接收缓冲器结束中断使能**
- **RXBUFF: 接收缓冲器满中断使能**

0 = 无效。

1 = 使能相应中断。

ADC 中断禁用寄存器

寄存器名称 :ADC_IDR

访问类型 :只写

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	RXBUFF	ENDRX	GOVRE	DRDY
15	14	13	12	11	10	9	8
OVRE7	OVRE6	OVRE5	OVRE4	OVRE3	OVRE2	OVRE1	OVRE0
7	6	5	4	3	2	1	0
EOC7	EOC6	EOC5	EOC4	EOC3	EOC2	EOC1	EOC0

- **EOCx: x 转换结束中断禁用**
- **OVREx: x 溢出错误中断禁用**
- **DRDY: 数据就绪中断禁用**
- **GOVRE: 通用溢出错误中断禁用**
- **ENDRX: 接收缓冲器结束中断禁用**
- **RXBUFF: 接收缓冲器满中断禁用**

0 = 无效。

1 = 禁用相应中断。

AT91SAM7S64 电气特性

绝对极限值

Table 86. 绝对极限值 *

工作温度 (工业级) ..... -40° C 到 +85° C
存储温度 ..... -60°C 到 +150°C
各个输入引脚对地电压..... -0.3V 到 +5.5V
最大工作电压 (VDDCORE 及 VDDPLL)..... 1.95V
最大工作电压 (VDDIO、 VDDIN 及 VDDFLASH)..... 3.6V
所有 I/O 线上直流输出电流 ..... 150 mA

*NOTICE: 如果强制芯片在超出“ 绝对极限值” 表中所列的条件之下工作可能造成器件的永久损坏。这仅是工作应力的极限。并不表示器件可以工作于表中所列条件之下，或是那些超越工作范围明确规定的其他条件之下。长时间工作于绝对极限值可能会影响器件的寿命。

直流特性

下列特性适用的温度范围：T_A = -40°C 到 85°C，除非另有说明，结温达到 T_J = 100°C。

Table 87. 直流特性

符号	参数	条件		最小值	典型值	最大值	单位
$V_{VDDCORE}$	内核直流电压			1.65		1.95	V
V_{VDDPLL}	PLL 直流电压			1.65		1.95	V
V_{VDDIO}	I/O 直流电压			3.0		3.6	V
$V_{VDDFLASH}$	Flash 直流电压			3.0		3.6	V
V_{IL}	输入低电平电压			-0.3		0.8	V
V_{IH}	输入高电平电压			2.0		5.5	V
V_{OL}	输出低电平电压	I_O = 8 mA				0.4	V
V_{OH}	输出高电平电压	I_O = 8 mA		V_{DDIO} - 0.4			V
I_{LEAK}	输入漏电流	上拉电阻禁用 (典型 : T_A = 25°C, 最大 : T_A = 85°C)			20	200	nA
I_{PULLUP}	输入上拉电流			143	321	600	µA
C_{IN}	输入电容	64-LQFP 封装				13.9	pF
I_{SC}	静电流	$V_{VDDCORE}$ = 1.85V, MCK = 0 Hz	T_A =25°C		26	50	µA
		所有输入驱动为 1 (包括 TMS, TDI, TCK, NRST)	T_A = 85°C		260	500	
I_O	输出电流	PA0-PA3				16	mA
		PA4-PA31				8	mA

Table 88. 1.8V 电压整流器特性

符号	参数	条件	最小值	典型值	最大值	单位
V_{VDDIN}	电源电压		3.0	3.3	3.6	V
V_{VDDOUT}	输出电压		1.81	1.85	1.89	V
I_{VDDIN}	电流消耗	启动后无负载		90		µA
		启动时无负载			100	mA
		空闲模式			20	µA
T_{START}	启动时间	V_{DDIN} > 2.7V 后 C_{load} = 2.2 µF			150	µS
I_O	最大直流输出电流	V_{DDIN} = 3.3V			100	mA
I_O	最大直流输出电流	V_{DDIN} = 3.3V, 空闲模式			1	mA

Table 89. 掉电检测器特性

符号	参数	条件		最小值	典型值	最大值
V_{BOT-}	门限电平		1.65	1.68	1.71	V
V_{HYST}	迟滞	$V_{HYST} = V_{BOT+} - V_{BOT-}$		50	65	mV
I_{DD}	电流消耗	BOD 开 (GPNVM0 位有效)		12	18	µA
		BOD 管 (GPNVM0 位无效)			1	mA
T_{START}	启动时间			100	200	ms

功耗

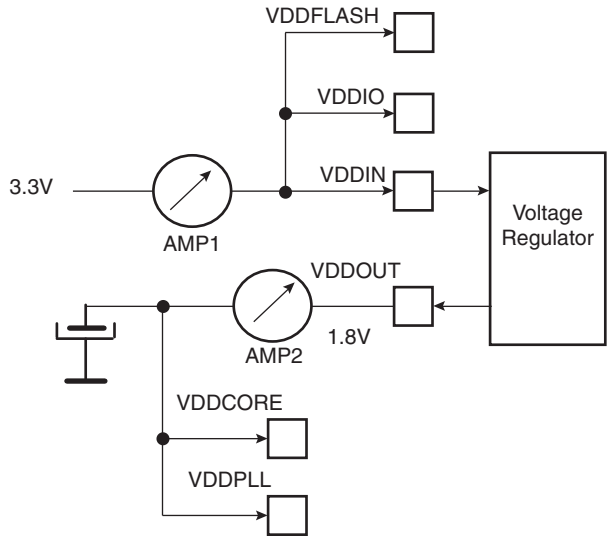
- PLL、慢时钟及主振荡器典型功耗。
- 两种不同供电模式功耗：激活与过低电源。
- 外设功耗：在使能然后禁用相应时钟后以不同电流值计算。

功耗与模式关系

Table 90 与 Table 91 on page 448 值估计如下工作条件下功耗：

- $V_{DDIO} = V_{DDIN} = V_{DDFLASH} = 3.3V$
- $V_{DDCORE} = V_{DDPLL} = 1.85V$
- $T_A = 25° C$
- MCK = 50 MHz
- I/O 上无消耗

Figure 188. 测量示意图



这些表给出功耗估计。

Table 90. 不同模式下功耗

模式	状态	消耗	单位
激活	电压整流器开。 掉电检测器激活。 读 Flash。 ARM 内核时钟为 50MHz。 模数转换器激活。 所有外设时钟工作。 在 AMP1 在 AMP2	 31.3 29.3	 mA
过低电源	电压整流器在低功耗模式。 掉电检测器无效。 Flash 处于 standby 模式。 ARM 内核时钟为 500Hz。 模数转换器无效。 所有外设时钟不工作。 在 AMP1 在 AMP2	 36.2 35.2	 µA

激活模式下外设功耗

Table 91. 外设功耗

外设	消耗	单位
PIO 控制器	0.4	mA
USART	0.9	
ADC	0.7	
PWM	0.3	
TWI	0.2	
SPI	0.9	
SSC	1.1	
定时器计数器通道	0.2	

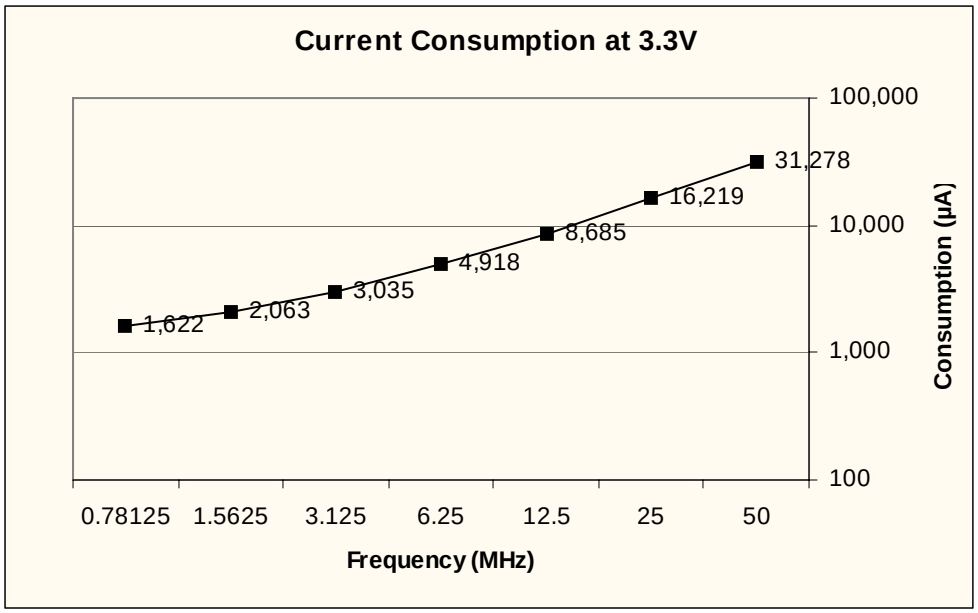
功耗与激活模式下主机时钟频率关系

Figure 189 给出以如下工作条件的估计值：

- $V_{DDIO} = V_{DDIN} = V_{DDFLASH} = 3.3V$
- $V_{DDCORE} = V_{DDPLL} = 1.85V$
- $T_A = 25° C$
- MCK 在 MHz 级
- 电压整流器开
- 掉电检测器激活
- 读 Flash
- 模数转换器激活
- 所有外设时钟工作
- I/O 上无消耗

Figure 189 给出估计功耗。

Figure 189. 功耗与激活模式下 MCK 频率的关系



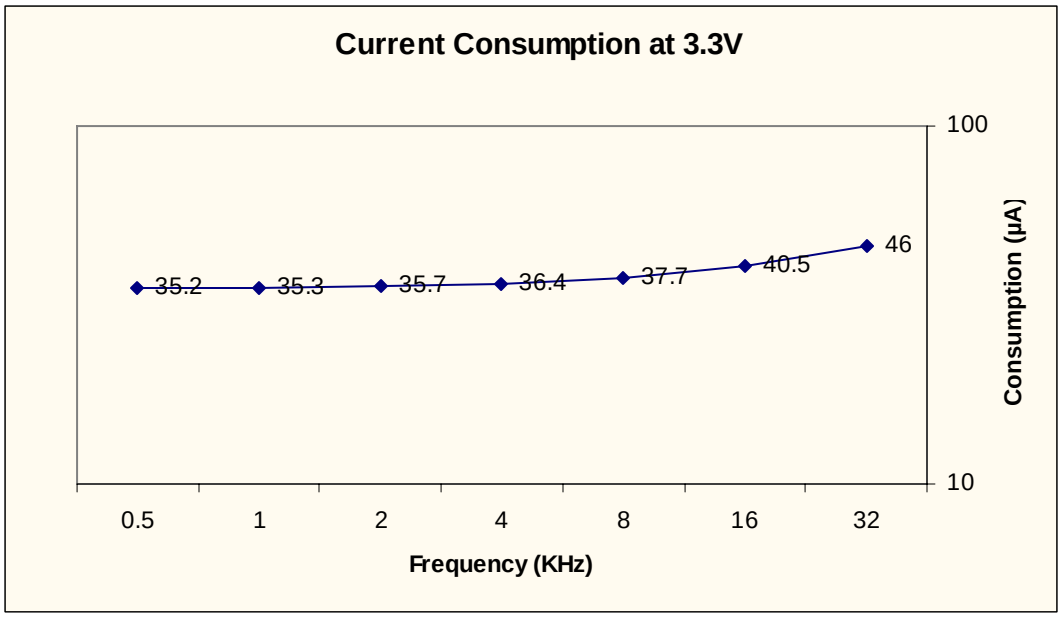
功耗与过低电压下主机时钟频率关系

Figure 190 给出以如下工作条件的估计值：

- $V_{DDIO} = V_{DDIN} = V_{DDFLASH} = 3.3V$
- $V_{DDCORE} = V_{DDPLL} = 1.85V$
- $T_A = 25° C$
- 电压整流器工作在低功耗模式
- 掉电检测器无效
- Flash 处于 standby 模式
- 模数转换器无效
- 所有外设时钟不工作
- PLL 等待
- 主振荡器等待
- I/O 上无消耗

Figure 190 给出估计功耗。

Figure 190. 功耗与过低电压模式下 MCK 频率关系



晶振特性

RC 振荡器特性

Table 92. RC 振荡器特性

符号	参数	状态	最小值	典型值	最大值	单位
$1/(t_{CPRC})$	RC 振荡器频率	V_{DDPLL} = 1.65V	22	32	42	KHz
	占空比		45	50	55	%
t_{ST}	启动时间	V_{DDPLL} = 1.65V			75	µs
I_{OSC}	电流消耗	启动后			1.9	µA

主振荡器特性

Table 93. 主振荡器特性

符号	参数	状态	最小值	典型值	最大值	单位
$1/(t_{CPMAIN})$	晶振频率		3	16	20	MHz
C_{L1}, C_{L2}	内部负载电容 ($C_{L1} = C_{L2}$)			25		pF
C_L	等效负载电容			12.5		pF
	占空比		40	50	60	%
t_{ST}	启动时间	V_{DDPLL} = 1.2 到 2V C_S = 3 pF(1) $1/(t_{CPMAIN})$ = 3 MHz C_S = 7 pF(1) $1/(t_{CPMAIN})$ = 16 MHz C_S = 7 pF(1) $1/(t_{CPMAIN})$ = 20 MHz			14.5 1.4 1	ms
I_{OSC}	电流消耗	激活模式			550	µA
		Standby 模式			1	µA

Notes: 1. C_S 为分流电容。

XIN 时钟特性

Table 94. XIN 时钟电气特性

符号	参数	状态	最小值	最大值	单位
$1/(t_{CPXIN})$	XIN 时钟频率			50.0	MHz
t_{CPXIN}	XIN 时钟周期		20.0		ns
t_{CHXIN}	XIN 时钟高半周期		0.4 x t_{CPXIN}	0.6 x t_{CPXIN}	
t_{CLXIN}	XIN 时钟低半周期		0.4 x t_{CPXIN}	0.6 x t_{CPXIN}	
C_{IN}	XIN 输入电容	(1)		25	pF
R_{IN}	XIN 下拉电阻	(1)		500	kΩ

Note: 1. 这些特性只有在主振荡器处于旁路模式时应用 (即当 CKGR_MOR 寄存器 MOSCEN = 0 且 OSCBYPASS = 1 , 见 “PMC Clock Generator Main Oscillator Register” on page 171)。

PLL 特性

Table 95. 锁相环特性

符号	参数	状态	最小值	典型值	最大值	单位
F_{OUT}	输出频率	CKGR_PLL 域出为：	00	80	160	MHz
			10	150	220	MHz
F_{IN}	输入频率			1	32	MHz
I_{PLL}	电流消耗	激活模式			4	mA
		Standby 模式			1	µA

Note: 启动时间由 PLL RC 滤波器确定。计算工具由 Atmel 提供。

ADC 特性

Table 96. 通道转换时间与 ADC 时钟

参数	状态	最小值	典型值	最大值	单位
ADC 时钟频率				5	MHz
启动时间	由空闲模式返回			20	µs
跟踪与保持获取时间		600			ns
转换时间	ADC 时钟 = 5 MHz			2	µs
吞吐率	ADC 时钟 = 5 MHz			384	kSPS

Table 97. 外部电压参考输入

参数	条件	最小值	最大值	单位
ADVREF 输入电压范围		2.6	V_{DDIN}	V
ADVREF 平均电流	在 13 采样 ADC 时钟 = 5 MHz	12	250	µA

Table 98. 模拟输入

参数	最小值	典型值	最大值	单位
输入电压范围	0		V_{ADVREF}	
输入漏电流		1		µA
输入电容		12	14	pF

Table 99. 发送特性

参数	条件	最小值	典型值	最大值	单位
分辨率			10		比特
整数非线性				±2	LSB
	ADC 时钟 = 5 MHz			±3	LSB
差分非线性				±1	LSB
	ADC 时钟 = 5 MHz			±2	LSB
偏移误差				±2	LSB
增益误差				±2	LSB

AT91SAM7S64 交流特性

适用条件及降额数据

这些条件及降额处理应用在以下章节中：时钟特性、内置 Flash 特性及 JTAG/ICE 定时。

条件与定时计算

所有给出的延迟是下列条件下的典型值：

- V_{DDIO} = 3.3V
- V_{DDCORE} = 1.8V
- 环境温度 = 25°C
- 负载电容 = 0 pF
- 输出电平变化检测为 (0.5 x V_{DDIO})。
- 低电平检测时输入电平为 0.8V ；高电平检测输入电平为 2.0V。

交流特性表中给出的最小最大值考虑了加工变化及设计。为获得其它条件下定时，使用下列等式：

$$t = \delta_{T^\circ} \times \left((\delta_{VDDCORE} \times t_{DATASHEET}) + \left(\delta_{VDDIO} \times \sum C_{SIGNAL} \times \delta_{CSIGNAL} \right) \right)$$

其中：

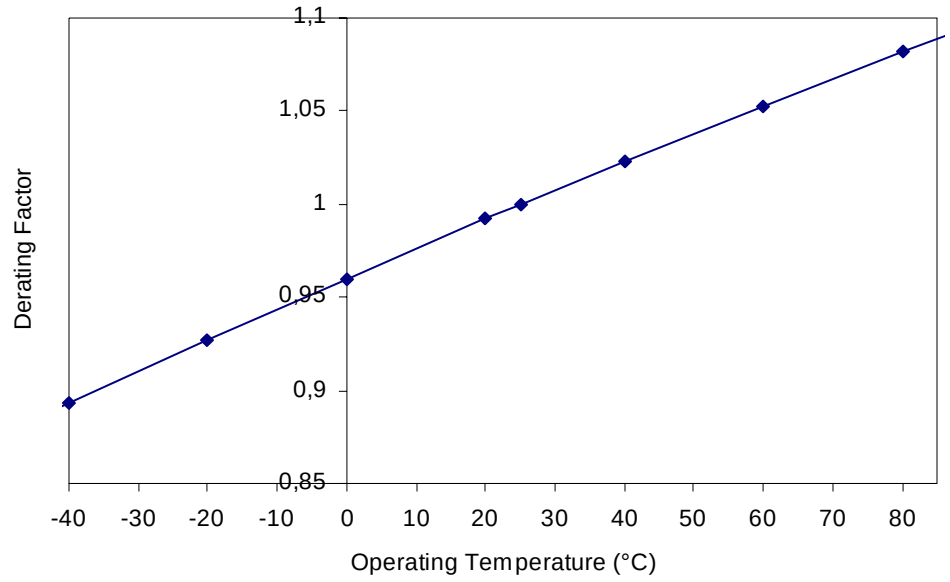
- δ_{T° 为温度降额因子，见 Figure 191 on page 455。
- $\delta_{VDDCORE}$ 为内核电源降额因子，见 Figure 192 on page 455。
- $t_{DATASHEET}$ 为负载电容为 0 pF 时的最小或最大定时值。
- δ_{VDDIO} 为 IO 电源降额因子，见 Figure 193 on page 456。
- C_{SIGNAL} 为输出引脚容性负载 (1)。
- $\delta_{CSIGNAL}$ 为根据手册中相关输出引脚容性负载最小或最大值的降额因子。

输入延迟为典型值。

Note: 用户必须考虑封装负载贡献 (C_{IN})，见 Table 87, “ 直流特性 ,” on page 444。

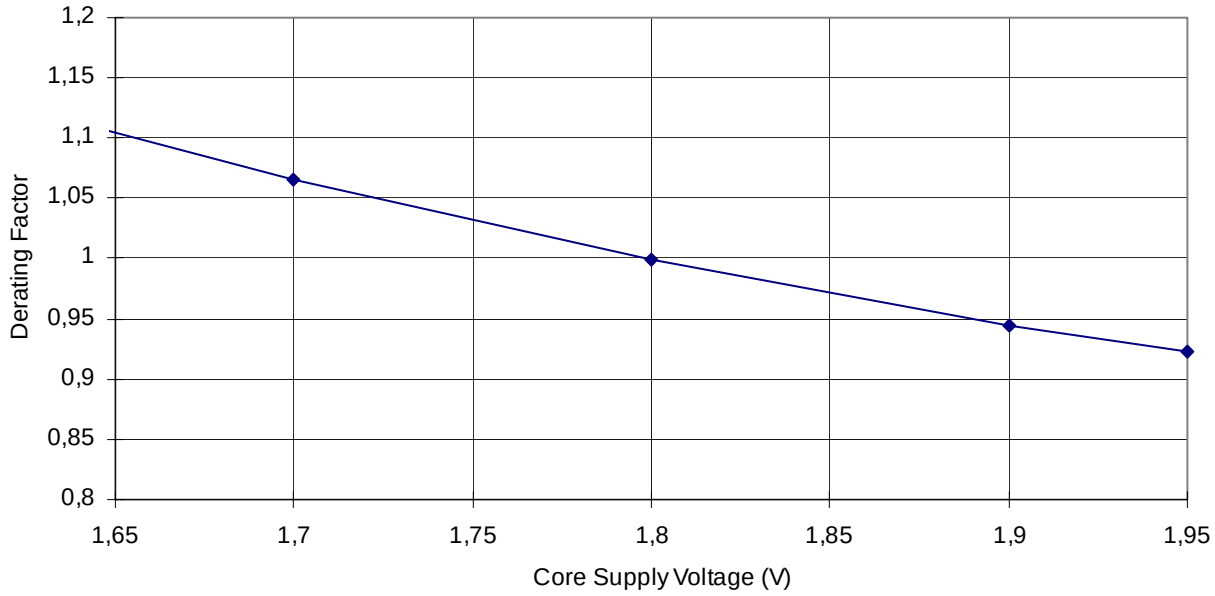
温度降额因子

Figure 191. 不同工作温度下降额曲线



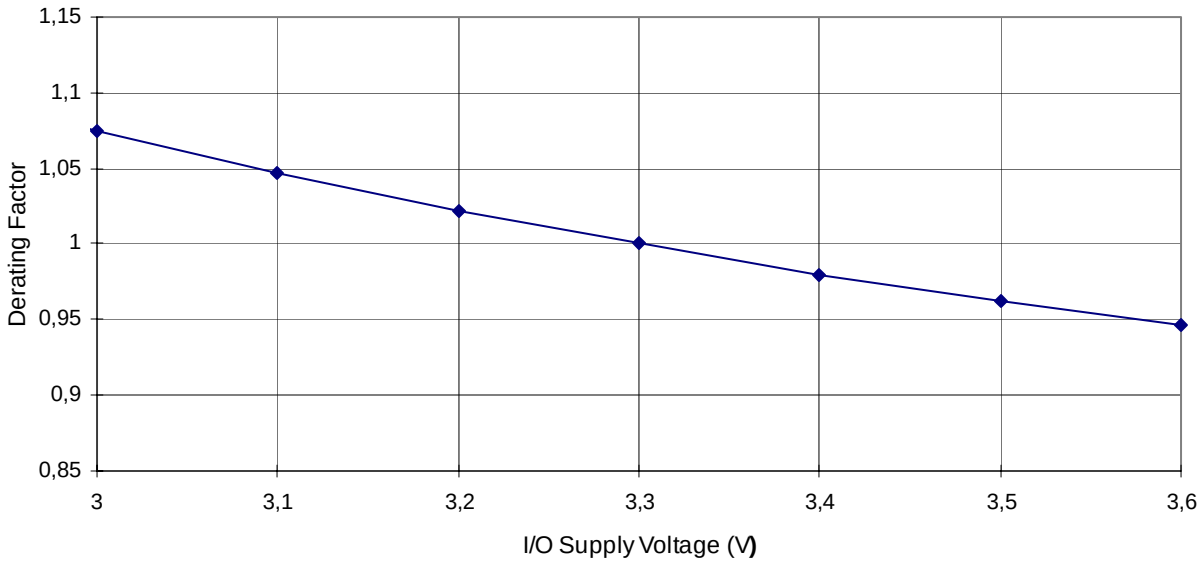
V_{DDCORE} 电压降额因子

Figure 192. 不同内核电压下降额因子



V_{DDIO} 电压降额因子

Figure 193. 不同 IO 电压下降额曲线



Note: 本例中降额因子仅适用于对相关输出引脚定时。

时钟特性

这些参数在下列条件下给出：

- V_{DDCORE} = 1.8V
- 环境温度 = 25°C

温度降额因子见 “ 适用条件及降额数据 ” on page 454 及 “VDDCORE 电压降额因子 ” on page 455。

主机时钟特性

Table 100. 主机时钟波形参数

符号	参数	条件	最小值	最大值	单位
$1/(t_{CPMCK})$	主机时钟频率			73	MHz

内置 Flash 特性

Table 101. 直流 Flash 特性

符号	参数	条件	最小值	最大值	单位
T_{PU}	上电延迟			30	µS
I_{SB}	Standby 电流	@85°C 在 VDDCORE = 1.8V 在 VDDFLASH = 3.3V		 0 30	µA
I_{CC}	激活电流	随机读 @ 40MHz 在 VDDCORE = 1.8V 在 VDDFLASH = 3.3V		 5.0 1.0	mA
		写 在 VDDCORE = 1.8V 在 VDDFLASH = 3.3V		 500 8.0	µA mA

Table 101 中给出最大工作频率 , 当处理器取指时受到内置Flash访问时间的限制。Table 102 给出根据MC_FMR寄存器FWS域值的器件最大工作频率。该域定义访问内置 Flash 存储器时所需的等待状态数。

Table 102. 内置 Flash 等待状态

FWS	读操作	最大工作频率 (MHz)
0	1 周期	40
1	2 周期	$1/(t_{CPMCK})$
2	3 周期	$1/(t_{CPMCK})$
3	4 周期	$1/(t_{CPMCK})$

Table 103. 交流 Flash 特性

参数	条件	最小值	最大值	单位
编程周期	每页包括自动擦除		4	ms
	每页包括自动擦除		2	ms
全芯片擦除		10		ms

JTAG/ICE 定时

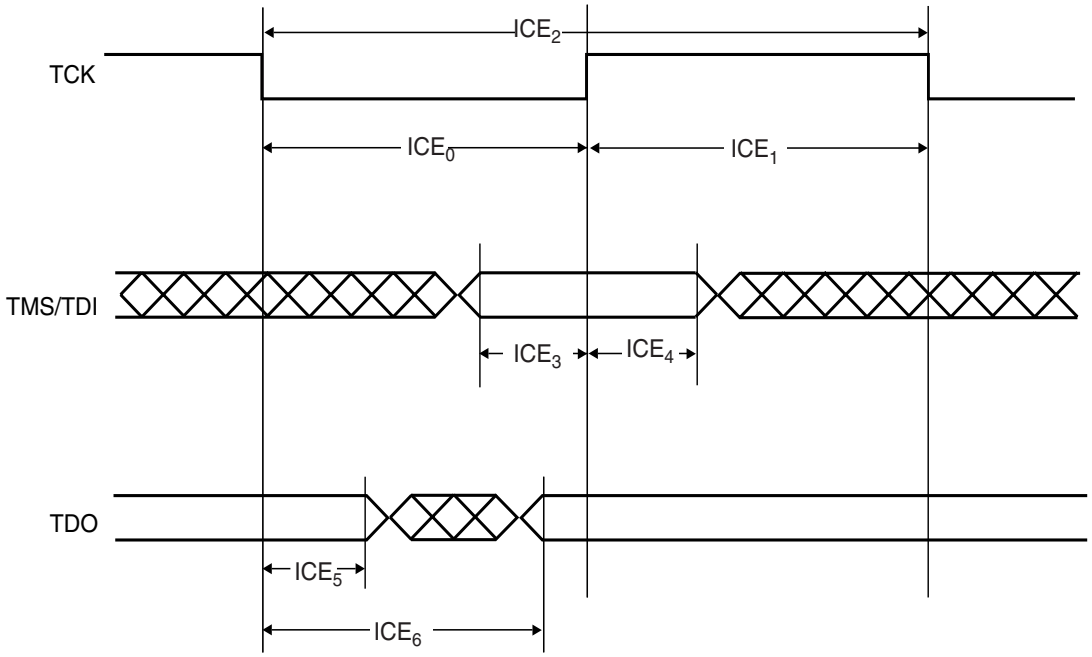
ICE 接口信号

Table 104 给出的定时的相关工作条件限制，见 “ 条件与定时计算 ” on page 454。

Table 104. ICE 接口定时规范

符号	参数	条件	最小值	最大值	单位
ICE_0	TCK 低半周期		51		ns
ICE_1	TCK 高半周期		51		ns
ICE_2	TCK 周期		102		ns
ICE_3	TCK 高前 TDI、 TMS 设置		3		ns
ICE_4	TCK 高后 TDI、 TMS 保持		0		ns
ICE_5	TDO 保持时间	C_{TDO} = 0 pF	3		ns
		C_{TDO} 降额	0.037		ns/pF
ICE_6	TCK 低到 TDO 有效	C_{TDO} = 0 pF		13	ns
		C_{TDO} 降额		0.037	ns/pF

Figure 194. ICE 接口信号



JTAG 接口信号

下表给出的定时的相关工作条件限制，见 “ 条件与定时计算 ” on page 454。

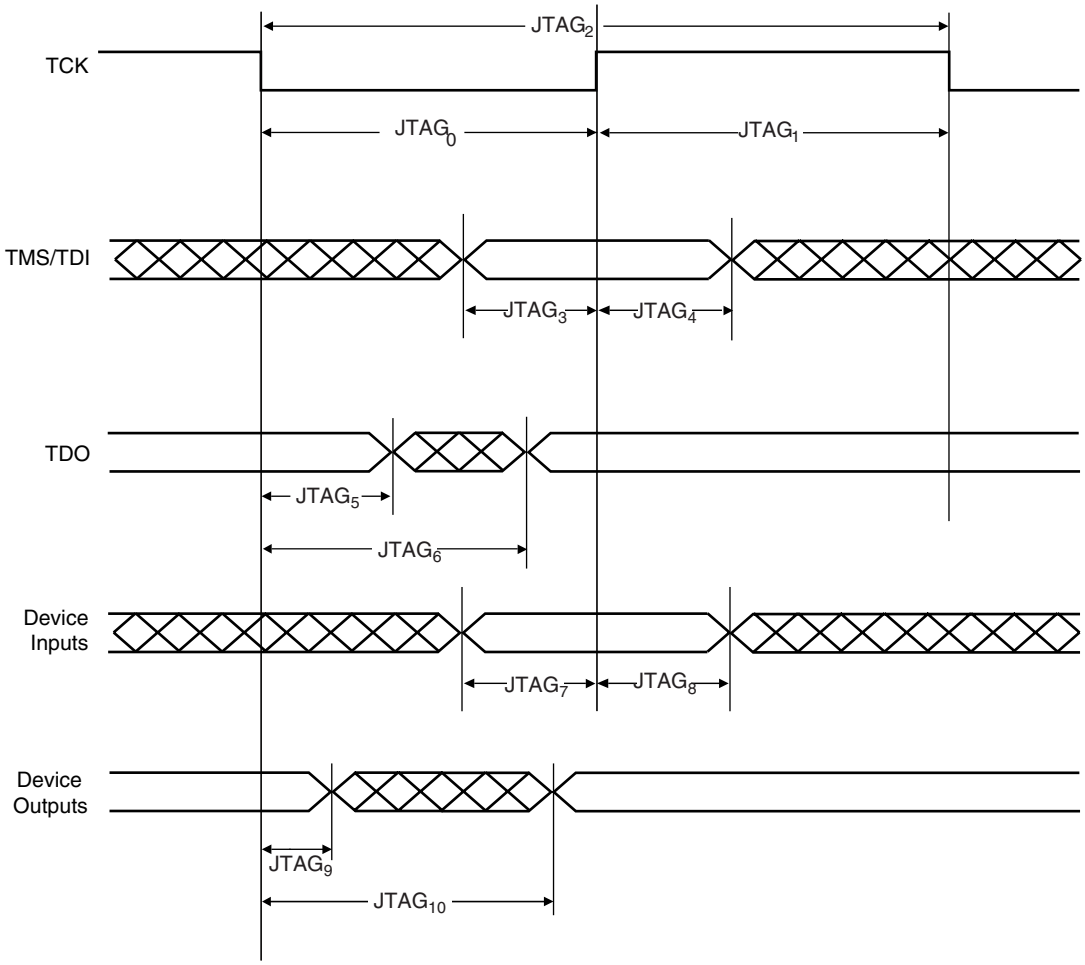
Table 105. JTAG 接口定时规范

符号	参数	条件	最小值	最大值	单位
$JTAG_0$	TCK 低半周期		6.5		ns
$JTAG_1$	TCK 高半周期		5.5		ns
$JTAG_2$	TCK 周期		12		ns

Table 105. JTAG 接口定时规范

符号	参数	条件	最小值	最大值	单位
$JTAG_3$	TCK 高前 TDI、 TMS 设置		2		ns
$JTAG_4$	TCK 高后 TDI、 TMS 保持		3		ns
$JTAG_5$	TDO 保持时间	C_{TDO} = 0 pF	2		ns
		C_{TDO} 降额	0.037		ns/pF
$JTAG_6$	TCK 低到 TDO 有效	C_{TDO} = 0 pF		15	ns
		C_{TDO} 降额		0.037	ns/pF
$JTAG_7$	器件输入设置时间		0		ns
$JTAG_8$	器件输入保持时间		3		ns
$JTAG_9$	器件输出保持时间	C_{OUT} = 0 pF	4		ns
		C_{OUT} 降额	0.037		ns/pF
$JTAG_{10}$	TCK 到器件输出有效	C_{OUT} = 0 pF		20	ns
		C_{OUT} 降额		0.037	ns/pF

Figure 195. JTAG 接口信号



AT91SAM7S64 机械特性

温度考虑

温度数据

Table 106中 ,器件寿命估计使用“适度控制”环境模型的MIL-217标准(该模型是指安置在一个有足够冷空气的固定架上) , 依靠器件结温 (详见 “ 结温 ” on page 463)。

注意使用 MTBF 计算时要非常小心。注意根据 MIL-217 模型得到的值为保守值 (在严格的条件在测试)。寿命测试结果往往比预取的要好。

Table 106. MTBF 与结温的关系

结温 (T_J) (°C)	估计寿命 (MTBF) (年)
100	17
125	9
150	5
175	3

Table 107 给出各封装下的热阻。

Table 107. 热阻数据

符号	参数	条件	封装	类型	单位
θ_{JA}	环境结温热阻	静态空气	LQFP64	47.2	°C/W
θ_{JC}	事件结温热阻		LQFP64	12.2	

结温

平均芯片结温 T_J, 可按下式得到，单位 °C：

4. $T_J = T_A + (P_D \times \theta_{JA})$
5. $T_J = T_A + (P_D \times (\theta_{HEATSINK} + \theta_{JC}))$

其中：

- θ_{JA} = 封装热阻，环境结温 (°C/W) 见 Table 107 on page 462。
- θ_{JC} = 封装热阻，事件结温热阻 (°C/W) 见 Table 107 on page 462。
- $\theta_{HEAT\ SINK}$ = 制冷器件热阻 (°C/W)，由器件手册给出。
- P_D = 器件功耗 (W) 估计，来自 “ 功耗 ” on page 446。
- T_A = 环境温度 (°C)。

根据第一个等式，可推导出芯片估计寿命，及决定是否需要制冷设备。若制冷设备不适合该芯片，使用第二个公式来计算平均芯片结温 T_J。

封装图

Figure 196. 64 引脚 LQFP 封装图

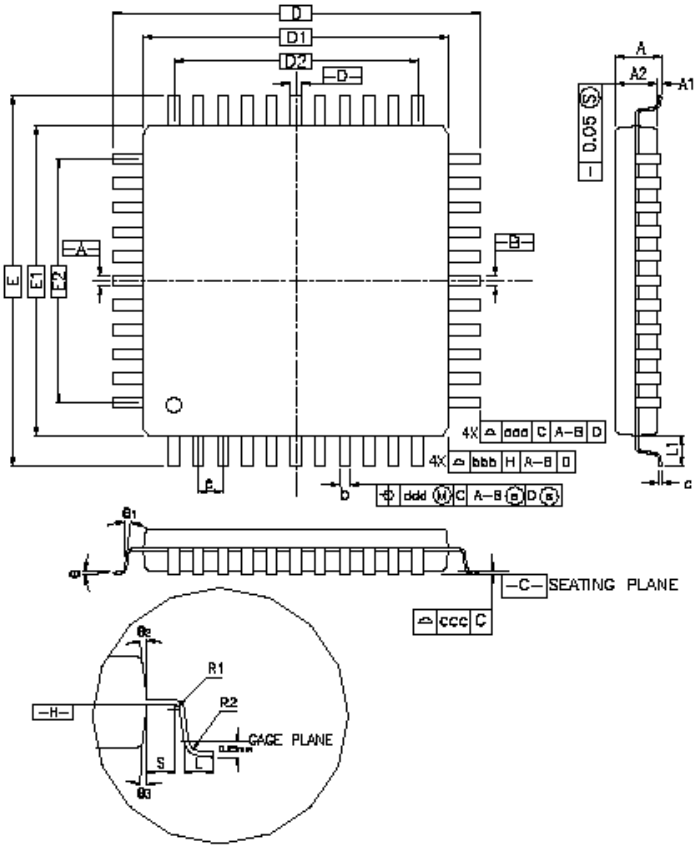


Table 108. 64 引脚 LQFP 封装尺寸 (单位：mm)

CONTROL DIMENSIONS ARE IN MILLIMETERS.

SYMBOL	MILLIMETER			INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	—	—	1.60	—	—	0.063
A_1	0.05	—	0.15	0.002	—	0.006
A_2	1.35	1.40	1.45	0.053	0.055	0.057
D	12.00 BSC.			0.472 BSC.		
D_1	10.00 BSC.			0.393 BSC.		
E	12.00 BSC.			0.472 BSC.		
E_1	10.00 BSC.			0.393 BSC.		
R_2	0.08	—	0.20	0.003	—	0.008
R_1	0.08	—	—	0.003	—	—
θ	0°	3.5°	7°	0°	3.5°	7°
θ_1	0°	—	—	0°	—	—
θ_R	11°	12°	13°	11°	12°	13°
θ_e	11°	12°	13°	11°	12°	13°
c	0.09	—	0.20	0.004	—	0.008
L	0.45	0.60	0.75	0.018	0.024	0.030
L_1	1.00 REF			0.039 REF		
S	0.20	—	—	0.008	—	—
b	0.17	0.20	0.27	0.007	0.008	0.011
e	0.50 BSC.			0.020 BSC.		
D2	7.50			0.295		
E2	7.50			0.295		
TOLERANCES OF FORM AND POSITION						
aaa	0.20			0.008		
bbb	0.20			0.008		
ccc	0.08			0.003		
ddd	0.08			0.003		

Table 109. 器件与 64 引脚 LQFP 封装最大重量

700	mg

Table 110. 64 引脚 LQFP 封装特性

湿度灵敏性级	3

焊接剖面图

Table 111 给出推荐的焊接剖面图， J-STD-20。

Table 111. 焊接剖面图

	对流或 IR/ 对流	VPR
平均上坡速率 (183° C 到峰值)	3° C/sec. max.	10° C/sec.
预热温度 125° C ±25° C	120 sec. max	
温度维持高于 183° C	60 sec. 到 150 sec.	
实际峰值温度 5° C 内时间	10 sec. 到 20 sec.	60 sec.
峰值温度范围	220 +5/-0° C 或 235 +5/-0° C	215 到 219° C 或 235 +5/-0° C
下坡速率	6° C/sec.	10° C/sec.
25° C 到峰值时间	6 min. max	

若板上大器件回流，小封装可能遇到高温度。此时，小封装抵挡的温度可能是235° C而非220° C (IR 回流)。

推荐的封装回流条件由封装密度及体积决定，见 Table 112。

Table 112. 推荐的封装回流条件 (1, 2, 3)

参数	温度
对流	235 +5/-0° C
VPR	235 +5/-0° C
IR/ 对流	235 +5/-0° C

当板上使用某些小封装时，这些小封装温度为 220°C，而非 235°C。

Notes:
1. 封装由 Atmel 使用 IR 回流条件，而非回流或 VPR，限定。
2. 默认情况下，封装级 1 限定在 220° C (除非规定为 235° C)。
3. 体温是最重要的参数但其它剖面参数如合计曝光时间到高温或加热速率等也可能影响元件可靠性。

每个元件允许三个最大的回流传输。

AT91SAM7S64 订货信息

Table 113. 订货信息

订货号	封装	温度 工作范围
AT91SAMS64-AI	LQFP 64	工业级 (-40°C 到 85°C)

Table of Contents

Revision History

Doc. Rev	Source	Comments [1]
Lit° 6070A		• **Date Qualified:** 28-Oct-04
		• **CSR's Implemented:** 19-Nov-04
	• CSR 04-204	• pg. 7, Table 2, pin 20 changed. • pg.107 -120 FFPI replaced with update to lit #6077
	• CSR 04-206	• pg. 3, Figure 1: Block Diagram. PGMM0-PGMM3
	• CSR 04-209	• pg. 106, change in PROGE description
	• CSR 04-210	• pg. 14; Security Bit Feature, Erase pin pull-down
	• CSR 04-220	• pg. 175, Change to CSS field description in PMC_PCKx register
	• CSR 04-231	• pg. 445, Table 89: Brownout Characteristics, table added (evolution of pagination)
	• CSR 04-232	• pg. 455, ADC Characteristics chapter inserted (evolution of pagination)
	• CSR 04-235	• pg. 452, Table 95, F_{OUT} 01 changed to 10 (evolution of page and table number)
	• CSR 04-241	• pg. 232, Table 54, NCPHA column changed
	• CSR 04-262	• pg. 14, modification to Flash Overview
	• CSR 04-270	• pg. 4, Table 1: Signal Description List, VDDPL function changed
	• CSR 04-273	• pg. 427, ADC; ccondition tags changed • Figure 185; VDDANA changed to VDDIN in ADC block diagram • GLobal: VDDANA removed.
	• CSR 04-274	• pg. 455, Table 99 and Table 100. changes
	• CSR 04-283	• pg. 444, Table 87: DC Characteristics, C_{IN} Max changed to 13.9
	• CSR 04-284	• pg. 452, Table 95: Phase Lock Loop Characteristics, Field 10, Max changed
	• CSR 04-292	• pg. 449, Figure 189, Chart replaced (pagination and figure # have evolved)
	• CSR 04-295	• pg. 10, I/O Line Drive Levels "...draw only 4 mA..." changed to..."draw only 8 mA..."
	• CSR 04-304	• pg. 26, Table 4: Peripheral Identifiers, changes to note 1.
	• CSR 04-323	• pg. 428, Conversion Performances, cross ref changed to ADC Characteristics
	• CSR 04-325	• pg. 103, change to FMCN
	• CSR 04-337	• pg. 445 Table 89, change to Brownout Detector Characteristics
	• CSR 04-338	• pg. 449, Chart replaced in Figure 189 (evolved from fig.187)
	• CSR 04- 339	• pg. 450, Chart replaced in Figure 190 (evolved from fig. 188)
	• CSR 04-342	• pg. 14, Lock Region information changed
	• CSR 04-343	• pg. 19, RC oscillator range changed
	• CSR 04-348	• pg. 4, Table 1; pg 8, Power Supplies; pg, 444, Table 88; VDDIN characteristics changed.
	• CSR 04-356	• pg. 14, Flash Overview modified
	• CSR 04-375	• pg. 169, PMC_SCSR register variable for UDP bit field implemented.
	• CSR 04-377	• pg. 174, PMC_MCKR bit field defined as "–"

Doc. Rev	Source	Comments [1]
6070A	• CSR 04-378	• pg. 456 AC Characteristics Chapter relocated
	• CSR 04-379	• pg. 460, Table 106: AC Flash Characteristics added
	• CSR 04-389	• pg. 457 - 458, Figures 192, 193, 194 derating curves modified

Doc. Rev	Source	Comments [1]
6070B		• **Date:** 16-Feb-05
		Global: Peripheral Data Controller (PDC) Changed to Peripheral DMA Controller (PDC)
	• CSR 05-0	• pg. 3, Figure 1, Block Diagram. PGMEN2 added • pg. 6, Table 1, Signal Description List. FFPI details, PGMEN2 added • pg. 7, Table 2, Pinout in 64-lead LQFP Package, PGMEN2 added and associated with PA2. • pg. 41, Table 10, JTAG Boundary Scan Register, PGMEN2 added and associated with PA2.
	• CSR 05-086	• pg. 14, Flash Overview, text added
	• CSR 05-081	• pg. 18, Reset Controller, text added
	• CSR 04-427	• pg. 19, PLL Output range changed • pg. 108, Table 19; pg 116, Table 33: Signal Description List, VDDPLL line changed
	• CSR 04-436	• pg. 40 Text relevant to Debug Unit removed from Functional Description
	• CSR 04-442	• pg. 40, Same as CSR 04-436
	• CSR 05-107	• pg. 91-92, MC_ASR Register. MST0 and MST1 definitions reversed. SVMST0 and SVMST1 definitions reversed.
	• CSR 04-440	• pg. 106, “disactive” replaced by “inactive”
	• CSR 04-451	• pg. 201, ARCH: Architecture Identifier evolution in DBGU_CIDR register
	• CSR 04-410	• pg. 381 note added below Figure 168. • pg. 382 Interrupts, section added.
	• CSR 04-411	• pg. 453, Restore USB Transeiver Characteristics
	• CSR 04-423	• pg. 468, AT91SAM7S64-AI Ordering Code corrected (superceded by CSR 05-010)
	Review	• pg. 450, Power Consumption versus Master CLock Frequency in Ultra Low-power Mode. “USB Transeiver de-activiated” added to list.
	• CSR 05-077	• pg. 442, Table 87: DC Characteristics, TSLOPE line added
	• CSR 05-072	• pg. 442, Table 88: 1.8 V Voltage Regulator Characteristics, modified
	• CSR 05-082	• pg. 423, Table 102: items removed.
	• CSR 05-075	• pg. 458, added to page, Table 107: Endurance Flash Characteristics
	Review	• pg. 460, Table 104: DC Flash Characteristics, @25° added to Standby Current, all Max values changed in table.
	• CSR 05-085	• pg. 464, Table 112, 64-lead-LQFP Package Dimensions, modified.
	• CSR 05-010	• pg. 468, Table 116, Ordering Information. Table replaced.

Note: 1. Due to additions and deletions of tables, figure drawings and changes to text, the pagination in the current document often does not correspond to that as stated in any given CSR. Under “comments” in the Revision History tables the numbering properties corresponding to changes made are therefore also approximate.

Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests
www.atmel.com/literature

6070A–ATARM–07-Jun-05